Basic-Texteditor

Das im letzten Heft abgedruckte Jahres-Inhaltsverzeichnis der FUNKSCHAU entstand mit dem im folgenden beschriebenem zeilenorientierten Texteditor, der sich für Verzeichnisse aller Art einsetzen läßt. Zum Abspeichern der Texte auf Kassette oder Floppy-Disk werden geeignete Routinen für die Computer PET-2001, CBM-3032 und AIM-65 bzw. PC-100 beschrieben.

Was tut ein Editor?

Als „Editor" bezeichnet man ein Programm, das das Einschreiben, Ändern und Auslesen von Textzeilen zuläßt. Bei den Textzeilen kann es sich um ein Inhaltsverzeichnis, eine Sammlung von Telefonnummern und Adressen oder wiederum ein Programm handeln, das später z.B. von einem Assembler übersetzt werden soll. Hier wird ein Editor beschrieben, der sich speziell für Verzeichnisse aller Art eignet: Er gestattet das zeilenweise Eingeben von Texten auch zwischen schon bestehende Zeilen, kann jede Zeile mit einer Rubriknummer versehen (Warengruppe u.a.) sowie nach diesen Rubriken und alphabetisch sortieren. Weitere Besonderheiten sind ein Stichwortsuchbefehl und der Ausdruck aller Zeilen mit einem vom Benutzer definierten Format, ohne Worte zu „zerreißen".

Ein paar Worte zum Programm

Das vollständige Editor-Programm ist in *Bild 1* aufgelistet. Es läuft unverändert auf den Computern PET-2001 und CBM-3032; für andere Systeme sind nur geringe Syntaxänderungen erforder-

```
10 PRINT"s********** E D I T O R **********":PRINT:REM VERSION2.1
20 INPUT"LINE WIDTH FOR PRINTOUT";W
30 D=150:REM MAX LINE NR
40 PRINT"SECTION NRS.? Y/N"
50 GETS$:IFS$=""GOTO50
60 DIMA$(D):DATAA,B,D,F,I,K,L,N,P,R,S,T,U
70 GOSUB950
80 PRINT"A=ALPHA SORT,B=BOTTOM,D=DOWN,F=FIND,"
90 PRINT"I=INSERT,K=KILL LINE,L=LOAD,"
100 PRINT"N=DEF.NR.,P=PRINT ALL,R=REMOVE NRS.,"
110 PRINT"S=SAVE FILE,T=TOP,U=UP ...?"
120 RESTORE:GETM$
130 FORI=1TO13:READB$:IFB$=M$GOTO160
140 NEXT
150 GOTO120
160 ONIGOSUB180,230,250,270,370,450,500,580,660,760,700,730,740
170 PRINTA$(J):GOTO120
180 S=0:I=-1:GETM$:IFM$=" "THENPRINT"SORT INTERRUPTED":RETURN
190 I=I+1:IFA$(I+1)="END"ANDS=0THENPRINT"SORTED":RETURN
200 IFA$(I+1)="END"GOTO180
210 IFA$(I)>A$(I+1)THENM$=A$(I):A$(I)=A$(I+1):A$(I+1)=M$:S=1
220 GOTO190
230 FORJ=0TOD:IFA$(J)="END"THENRETURN
240 NEXT
250 IFA$(J)<>"END"THENJ=J+1
260 RETURN
270 REM FIND WORD
280 J=-1
290 INPUT"WORD=";B$:K=LEN(B$)
300 J=J+1:IFA$(J)="END"THENPRINT:RETURN
310 S=LEN(A$(J))-K:IFS<0GOTO300
320 FORI=1TOS+1:M$=MID$(A$(J),I,K)
330 IFM$=B$GOTO360
340 NEXT
350 GOTO300
360 PRINTA$(J):GOTO300
370 REM INSERT
380 I=-1
390 I=I+1:IFI=D-1THENPRINT"MEMORY OVERFLOW":RETURN
400 IFA$(I)<>"END"GOTO390
410 A$(I+1)=A$(I):I=I-1:IFI>=JGOTO410
420 INPUTM$:IFM$="END"THENI=J:GOTO480
430 IFS$="Y"THENM$="** "+M$
440 A$(J)=M$:J=J+1:GOTO380
450 PRINT"KILL LINE? Y/N":I=J
460 GETM$:IFM$<"N"GOTO460
470 IFM$<>"Y"THENRETURN
480 I=I+1:IFA$(I-1)="END"THENRETURN
490 A$(I-1)=A$(I):GOTO480
500 PRINT"CLEAR LAST FILE? Y/N"
510 GETM$:IFM$=""GOTO510
520 IFM$="Y"THENGOSUB950
530 GOSUB230
540 OPEN15,8,15:INPUT"FILENAME=";M$:M$="1:"+M$:
    OPEN1,8,0,M$:PRINT"LOADING"
550 INPUT#1,A$(J):IFASC(A$(J))<33THENA$(J)=RIGHT$(A$(J),
    LEN(A$(J))-1)
560 IFA$(J)<>"END"THENJ=J+1:GOTO550
570 INPUT#15,M$,B$,C$,D$:PRINTM$,B$,C$,D$:CLOSE1:CLOSE15:
    PRINT"LOADED":RETURN
580 IFS$="Y"GOTO620
590 S$="Y"
600 FORI=0TOD:IFA$(I)="END"GOTO620
610 A$(I)="** "+A$(I):NEXT
620 IFA$(J)="END"THENRETURN:REM DEF NR
630 INPUT"RUBRIC=";M$:A$(J)=RIGHT$(A$(J),LEN(A$(J))-2)
640 IFLEN(M$)=1THENM$="#"+M$
650 A$(J)=M$+A$(J):S$="Y":GOTO250
660 PRINT:R$="":REM PRINT ALL
670 FORI=0TOD:IFA$(I)="END"THENPRINT:RETURN
680 IFA$(I)>""THENGOSUB820
690 NEXT
700 INPUT"FILENAME=";M$:OPEN1,1,1,M$:J=0
710 PRINT#1,A$(J):IFA$(J)<>"END"THENJ=J+1:GOTO710
720 CLOSE1:PRINT"SAVED":RETURN
730 J=0:RETURN
740 IFJ>0THENJ=J-1
750 RETURN
760 PRINT"CUT OFF NR.FIELD? Y/N"
770 GETM$:IFM$=""GOTO770
780 IFM$<>"Y"THENRETURN
790 S$="N":PRINT"DELETED"
800 FORI=0TOD:IFA$(I)="END"THENRETURN
810 A$(I)=RIGHT$(A$(I),LEN(A$(I))-3):NEXT
820 M$=A$(I):B$=LEFT$(M$,2):IFS$="N"GOTO850
830 IFB$<>R$THENR$=B$:PRINT:PRINT"R U B R I K  ";R$:PRINT
840 M$=RIGHT$(M$,LEN(M$)-3)
850 IFLEFT$(M$,1)=" "THENM$=RIGHT$(M$,LEN(M$)-1):GOTO850
860 IFW>LEN(M$)THENPRINTM$:RETURN
870 K=W+1
880 K=K-1:IFK=0THENK=W:GOTO920
890 B$=MID$(M$,K,1):IFB$>"?"GOTO880
900 IFB$<":"ANDB$>"/"GOTO880
910 IFB$="'"ORB$="("GOTO880
920 PRINTLEFT$(M$,K)
930 M$=RIGHT$(M$,LEN(M$)-K):PRINT" ";
940 GOTO850
950 FORI=0TOD:A$(I)="":NEXT
960 J=0:A$(0)="END":PRINT"FILE CLEARED":RETURN
READY.
```

Bild 1. Das komplette Editor-Programm in der PET/CBM-Syntax mit den Routinen für das Abspeichern von Files auf Kassette. Das kleine „s" im Print-Befehl von Zeile 10 löscht den Bildschirm

lich. Unbedingte Voraussetzung ist allerdings das Vorhandensein der verwendeten Stringbefehle. Beim TRS-80 (Level II) und beim ABC-80 ist eine Vereinfachung des Programmteils für die Stichwortsuche durch den Befehl INSTR (Instring) möglich. Das Problem der Kleinbuchstaben-Verarbeitung ist sehr systemabhängig und wird daher hier nicht näher beschrieben.

Jeder Befehl besteht nur aus einem einzigen Buchstaben (*Tabelle*); es ist nicht erforderlich, danach die Return- bzw. Enter-Taste zu drücken. Zu bedenken ist aber, daß die Tastatur mit dem GET-Befehl abgefragt wird, was die Verwendung eines seriellen Terminals (z.B. TTY) leider ausschließt. Nach der Ausführung eines Befehls wird – damit man weiß, wo man sich gerade im Text befindet – die aktuelle Zeile angezeigt.

Als „File" bezeichnet man die Gesamtheit der gespeicherten Textzeilen. Sie wird hier durch eine eindimensionale String-Variable A$(J) repräsentiert, wobei J die Zeilennummer darstellt. Die Länge der Zeilen ist durch die Länge des für die Eingabe nun einmal erforderlichen Input-Buffers des Computersystems begrenzt (meist 60...80 Zeichen) Da er beim PET im Video-RAM liegt, wird manchmal das Fragezeichen beim Input-Befehl mitgespeichert. Mit der Zeile

425 IF LEFT (M$,1) = "?" OR LEFT (M$,1) = " " THEN M$ = MID (M$,2):

GOTO 425 ist hier aber leicht Abhilfe zu schaffen.

Systemabhängig sind in erster Linie die Routinen für das Abspeichern auf und das Laden vom Band (bzw. Disk). Die in Bild 1 vorhandenen Routinen gelten für die Computer PET-2001 und CBM-3032. Dabei wird auch die letzte, immer „END" lautende Textzeile mitgespeichert, die das Ende der File markiert.

Der unvermeidliche Nachteil aller Programme, die auf einem auf Interpreter-Basis Computer laufen, haftet natürlich auch diesem Editor-Programm an: Es ist wesentlich langsamer als Maschinensprache-Programme [1]. So werden etwa 30 Sekunden benötigt, um eine File von 30 unsortierten Zeilen nach dem Alphabet zu ordnen, und dieser Zeitbedarf steigt mit wachsender Zeilenzahl unverhältnismäßig schnell an. Die Stichwortsuchzeit beträgt bei einem typischen 30-Zeilen-File rund 10 Sekunden.

Der Umgang mit dem Editor

Die Tabelle erläutert die Wirkung der unterschiedlichen Systembefehle, die, wie erwähnt, nur aus einem Buchstaben bestehen. An mehreren Stellen wartet das System jedoch auf Benutzereingaben, die mit der Return-Taste abzuschließen sind, nämlich die gewünschte Zeilenbreite, die Textzeilen selbst, den File-Namen beim Laden und Abspeichern sowie die Rubriknummer. An diesen Stellen ist besondere Vorsicht geboten: Drückt man sofort die Return-Taste, ohne etwas eingegeben zu haben, so wird das Editor-Programm verlassen, und man ist im Basic-Direktmodus. Auf keinen Fall darf man den Editor jetzt wieder mit RUN starten, da sonst alle Textzeilen verloren sind (RUN setzt die Variablenpointer des Basic-Interpreters neu). Vielmehr kann das Programm ohne Datenverlust mit CONT (Continue) fortgesetzt werden, wobei man wieder zur Eingabefunktion gelangt, die man versehentlich verlassen hatte. An jeder anderen Stelle kann man das Programm nur mit der Break- bzw. Stop-Taste verlassen.

Das Abspeichern auf Kassette mit Taste „S" geschieht mit einem frei wählbaren Namen, ebenso das Wiedereinlesen: Gelesen wird nur eine File, deren Namen mit dem nach „L" eingegebenen übereinstimmt. Dabei kann man noch wählen, ob die alte File gelöscht oder die neue an die alte angehängt werden soll. Das Anhängen einer File mit Rubriknummern an eine ohne Rubriknummern (oder umgekehrt) ist unzulässig, da sich sonst Formatprobleme ergeben.

Die Rubriknummer

Die Rubriknummer (Section Nr.) ist eine maximal zweistellige Zahl, die einer Textzeile vorangestellt werden kann. Alle Zeilen, für die keine Rubrik definiert ist, beginnen mit „**", um auf das Fehlen der Nummer aufmerksam zu machen. Um der zuletzt angezeigten Zeile eine Rubrik zuzuweisen, drückt man N und gibt die jeweilige Nummer ein. Im FUNKSCHAU-Inhaltsverzeichnis haben z. B. alle Artikel über Meßgeräte eine gemeinsame gleiche Rubriknummer; eine andere Nummer ist für „Microcomputer" vorgesehen, usw. Auch ein bis zwei Buchstaben sind als „Nummer" möglich.

Sind alle Zeilen mit einer Nummer versehen und sortiert man sie nun mit Taste A alphabetisch durch, so wird zunächst nach Rubriken und innerhalb der Rubriken alphabetisch geordnet, wie das auch im Inhaltsverzeichnis von Heft 26/1979 zu sehen war. Will man dagegen ohne Rücksicht auf Rubriken editieren und sortieren, so kann man mit Taste R die ersten drei Zeichenfelder aller Zeilen der File und damit die Rubriknummern abschneiden; sie sind dann allerdings verloren (es sei denn, man hat die File vorher auf Kassette oder Disk abgespeichert). Beim Drükken von N werden sofort wieder die Rubriknummernfelder reserviert und mit „**" belegt.

Befehlsvorrat des Editors

Taste	*Befehl*	*Wirkung*
A	Alpha Sort	Sortieren der Zeilen nach Rubriknummern (falls vorhanden) und alphabetisch; Unterbrechen durch längeres Drücken der Leertaste (Space)
B	Bottom	Letzte Zeile anzeigen (immer END)
D	Down	Eine Zeile nach unten gehen
F	Find	In allen Zeilen nach einem Stichwort suchen
I	Insert[1])	Neue Zeile(n) vor die zuletzt angezeigte einfügen (mit END abschließen)
K	Kill	Angezeigte Zeile löschen (die folgenden Zeilen rücken nach)
L	Load	File von der Kassette oder Disk laden
N	Number	Angezeigte Zeile mit Rubriknummer versehen (nur bei END nicht möglich)
P	Print all	Alle Zeilen formatiert ausdrucken; falls Rubriknummern vorhanden, mit diesen
R	Remove number	Das Nummernfeld, d.h. die ersten drei Zeichen jeder Zeile, abtrennen
S	Save	File auf Kassette oder Disk speichern
T	Top	Erste („oberste") Textzeile anzeigen
U	Up	Eine Zeile nach oben gehen

[1]) Zeilen, die Trennungszeichen enthalten (Kommas oder Doppelpunkte), sind bei der Eingabe in Anführungszeichen zu setzen. Die Zeile selbst darf solche Anführungszeichen dann nicht enthalten; eine Alternative ist die Verwendung des Apostroph-Zeichens (Shift 7 auf der Tastatur).

```
540 POKE188,72:POKE189,232:I=ATN(0)
550 INPUTA$(J):IFA$(J)<>"END"THENJ=J+1:GOTO550
560 POKE188,3:POKE189,255:I=ATN(0):PRINT"LOADED":RETURN

700 J=0:POKE188,113:POKE189,232:POKE41993,32:I=ATN(0)
710 PRINT A$(J):IFA$(J)<>"END"THENJ=J+1:GOTO710
720 PRINTCHR$(13):POKE188,10:POKE189,229:I=ATN(0):
    PRINT"SAVED":RETURN
```

Bild 2. Load- und Save-Routinen für die Mikrocomputer AIM-65 und PC-100, die keine File-Befehle in Basic kennen. Im übrigen Editorprogramm müssen alle Print-Befehle ohne Argument durch PRINT" " ersetzt werden, um einen Zeilenvorschub zu bewirken

Formatierter Ausdruck der Zeilen

Mit P kann die File mit der nach dem Programmstart definierten Zeilenlänge aufgelistet werden. Dabei erfolgt die Ausgabe so, daß keine Worte getrennt werden; d.h. eine neue Zeile wird nur nach Erkennen eines Leerraums, eines Bindestrichs, eines Kommas oder ähnlicher Zeichen begonnen. Wird eine Zeile beim Ausdruck auf mehrere Druckzeilen verteilt, so werden nach der ersten Druckzeile alle folgenden, zur gespeicherten Zeile gehörigen um ein Zeichenfeld eingerückt, um den Ausdruck übersichtlich zu gestalten. Sollte eine Zeile an der Trennungsstelle mehrere Leerräume hintereinander enthalten, so werden die folgenden Leerräume bis zum ersten Buchstaben ignoriert, um das Format nicht zu stören. Die für die Ansteuerung eines externen Druckers notwendigen Befehle wurden hier nicht wiedergegeben, da sie vom Druckertyp und vom Computersystem abhängen. Bei Bedarf lassen sich die dafür erforderlichen Befehle OPEN und PRINT in der Druckroutine (Zeilen 660...690 und 820...940) unterbringen.

Load- und Save-Routinen

In Bild 1 dienen die Zeilen 540...570 dem Load- und 700...720 dem Save-Befehl. Sie sind für die Verwendung des im PET-2001 eingebauten Kassettenlaufwerks ausgelegt.

Beim AIM-65 bzw. PC-100 sind die Routinen in *Bild 2* zu verwenden. Diese beiden Computer verfügen über keinerlei File-Abspeicherungsbefehle in Basic, so daß mit Hilfe der ATN-Blindfunktion Maschinenprogramme für das Öffnen und Schließen von Files verwendet werden. Bei geöffneter Kassettenfile beziehen sich INPUT- und PRINT-Befehle auf den Kassettenrecorder. Bei der Load-Routine ist es u. U. sinnvoll, durch Einfügen des Befehls POKE 42001,0 den Drucker abzuschalten und bei Bedarf später mit POKE 42001,128 wieder zu aktivieren. Die Save-Routine setzt in Zeile 700 mit POKE 41993,32 den Gap-Wert in A409 (Pause zwischen den Datenblöcken) auf dezimal 32 bzw. hex 20; bei längeren Files kann das jedoch problematisch werden. Dann sollte man den Kassettenrecorder fernsteuern, wie das im System-Handbuch beschrieben ist.

Das Abspeichern von Files auf Diskette ist die eleganteste Methode. *Bild 3* zeigt die bei der CBM-3032-Floppy notwendigen Routinen für das linke Laufwerk (Nr. 1). Das Programm benutzt den Datenkanal 8 und den Fehlermeldungs-Kanal 15 des IEC-Bus. Beim Laden beseitigt es auch das von der Floppy-Disk an den Anfang jeder Zeile hinzugefügte Line-Feed-Zeichen [2]. Geeignete Floppy-Routinen für den TRS-80 sind in [3] und [4] beschrieben.

Herwig Feichtinger

Literatur

[1] *Feichtinger, H.*: KIM auf Datensuche. FUNKSCHAU 1978, Heft 24, Seite 1222 (siehe auch „Anwendungsbeispiele für den Mikroprozessor 6502" RPB 173, Seite 69).

[2] *Joepgen, H.G.*: Unsichtbares mischt mit – Über die Tücken nichtdruckender ASCII-Zeichen. FUNKSCHAU-Sonderheft HOBBYCOMPUTER 2, Seite 38.

[3] *Hofer, R.*: Dateiverarbeitung mit dem TRS-80. FUNKSCHAU 1979, Heft 15, Seite 887.

[4] *Hofer, R.*: Datenbank-Programm für den TRS-80. FUNKSCHAU 1979, Heft 16, Seite 955.

Stichworte zum Inhalt

Editor, Stichwortsuche, Files, Sortieren, Floppy-Disk, PET-2001, CBM-3032, AIM-65, PC-100.

Tiny-Basic-Ausdruck mit Baudot-TTY

Bei dem 2-KByte-Tiny-Basic für den KIM-1 läßt sich zusammen mit dem in Heft 16/1979 auf Seite 958 abgedruckten Baudot-Ausgabe-Unterprogramm recht einfach der Ausdruck aller auf dem Terminal eingetippten und vom Computer ausgegebenen Texte und Zahlen verwirklichen. Bei der Basic-Version, die bei der Adresse (hex) 2000 beginnt, sind dabei Sprünge zu Eingabe/Ausgabe-Routinen zu programmieren, die die Baudot-Ausgabe mit einbeziehen. Insgesamt sind folgende Änderungen und Erweiterungen nötig:

```
01B0 20 5A 1E    JSR 1E5A
01B3 48          PHA
01B4 C9 0A       CMP #0A
01B6 30 07       BMI 07
01B8 C9 5B       CMP#5B
01BA 10 03       BPL 03
01BC 20 00 01    JSR 0100
01BF 68          PLA
01C0 60          RTS
01C1 20 B3 01    JSR 01B3
01C4 4C A0 1E    JMP 1EA0
2006 4C B0 01    JMP 01B0
2009 4C C1 01    JMP 01C1
```

Bei der Tiny-Basic-Version mit Startadresse 0200 stehen die Sprungbefehle zur Ein-/Ausgabe entsprechend bei 0206 und 0209. Der Baudot-Fernschreiber druckt alles aus, was auch auf dem Terminal-Bildschirm sichtbar ist; verständlicherweise ist „Back Space" nicht möglich. Mit dem LIST-Befehl kann man ein gespeichertes Programm auf den Fernschreiber ausdrucken; Einschränkungen bestehen nur hinsichtlich des Baudot-Zeichenvorrats. Fe.

```
540 OPEN15,8,15:INPUT"FILENAME=";M$:M$="1:"+M$:OPEN1,8,0,M$:PRINT"LOADING"
550 INPUT#1,A$(J):IFASC(A$(J))<33THENA$(J)=RIGHT$(A$(J),LEN(A$(J))-1)
560 IFA$(J)<>"END"THENJ=J+1:GOTO550
570 INPUT#15,M$,B$,C$,D$:PRINTM$,B$,C$,D$:CLOSE1:CLOSE15:PRINT"LOADED":RETURN

700 INPUT"FILENAME=";M$:M$="1:"+M$:OPEN1,8,1,M$:J=0:OPEN15,8,15
710 PRINT#1,A$(J):IFA$(J)<>"END"THENJ=J+1:GOTO710
720 INPUT#15,M$,B$,C$,D$:PRINTM$,B$,C$,D$:CLOSE1:CLOSE15:PRINT"SAVED":RETURN
```

Bild 3. Routinen für die Verwendung der Dual-Floppy beim CBM-3032. Weitere Änderungen gegenüber Bild 1 sind hier nicht erforderlich

BASIC mit dem AIM-65

Für den bereits in Heft 10/1979 vorgestellten Mikrocomputer AIM-65, der praktisch mit dem Siemens-Computer PC-100 identisch ist, ist ein recht brauchbares 8-KByte-BASIC in Form zweier ROMs zu haben. Es wurde von Microsoft entwickelt und entspricht in vielen Punkten dem vom PET bekannten BASIC, unterscheidet sich aber in mehreren erwähnenswerten Dingen von anderen BASIC-Interpretern.

Die *Tabelle* zeigt den Befehlsvorrat. Die Befehle haben z. T. eine ungewohnte Wirkung, wie aus folgender Übersicht hervorgeht:

PRINT bewirkt ohne Argument keinen Zeilenvorschub. Will man in die nächste Zeile vorrücken, so muß man Print" " eingeben. Bei Verwendung des AIM-Displays sollte man auszugebende Texte mit einem Leerraum beginnen lassen, da der Cursor das erste (linke) Zeichen im Display überschreibt, sobald der Computer in den Eingabemodus springt; längere Texte sollte man dann außerdem zeilenweise mit einer kleinen Verzögerungsschleife ausgeben (z. B. FOR I = 1 TO 2000 : NEXT), da die Texte sonst nur ganz kurz auf dem Display aufblitzen.

ATN(A) wird zwar als Funktion erkannt; das notwendige Maschinenprogramm dafür paßte aber offenbar nicht mehr in die 8-KByte-ROM und muß daher, wenn diese Funktion gebraucht wird, vorher ins RAM geschrieben werden. Das Handbuch enthält das notwendige kurze Programm.

Files gibt es beim AIM-65 im Prinzip nicht, deswegen fehlen auch die Befehle OPEN und CLOSE. Nützlich ist dagegen die Möglichkeit, mit dem LOAD-Befehl Programme von der Kassette einzulesen, ohne bestehende zu zerstören; einzige Bedingung ist, daß unterschiedliche Zeilennummern verwendet werden. Dies ermöglicht z. B. das Einlesen unterschiedlicher DATA-Blöcke für ein Karteiprogramm – eine kleine Entschädigung für das Fehlen von Files.

Mit A=USR(B) kann man ein Maschinenprogramm aufrufen, das der Variablen A einen aus B errechneten Wert zuweist. Die Anfangsadresse dieses Maschinenprogramms muß vorher mit POKE in die Zeilen 0004 und 0005 gespeichert worden sein.

CLEAR löscht hier nicht das Programm, sondern ermöglicht es, in einem laufenden Programm alle Variablen, die READ-DATA-Zeiger und die FOR-NEXT-Zähler rückzusetzen. Mit WAIT(A,B,C) kann man das Programm warten lassen, bis die mit B maskierten Bits des I/O-Ports an der dezimalen Adresse A den Zustand C annehmen; läßt man C weg, wird dafür Null angenommen.

POS(0) liefert die derzeitige Cursor-Position einer Zeile, wobei (0) ebenso wie bei FRE zwar notwendig ist, aber ignoriert wird (dummy argument).

SAVE und LOAD dienen der Speicherung von Programmen mit einem Kassettenrecorder. Aus dem oben schon erwähnten Grund ist es sinnvoll, vor LOAD noch NEW einzugeben, um ein Vermischen von Programmen zu vermeiden. Als Programmname können max. fünf Zeichen dienen.

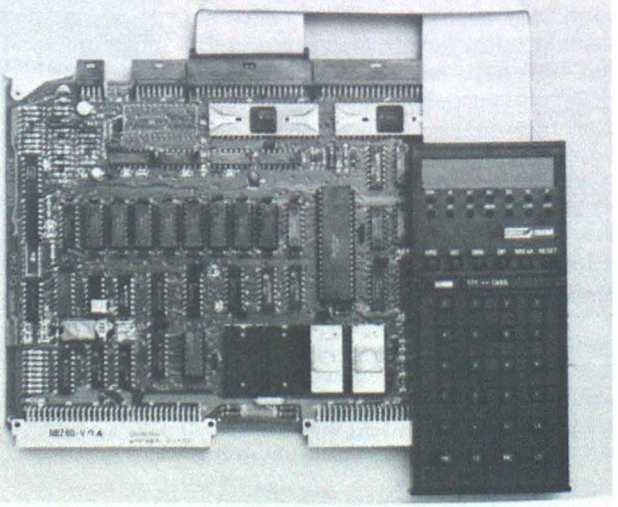
Mit der Taste F1 kann man ein laufendes Programm unterbrechen (Break). Der Befehl GET A$ weist der String-Variablen A den Wert einer gerade gedrückten AIM-Taste zu, aber – im Gegensatz zu INPUT – ohne das Programm anzuhalten, also ebenso wie beim PET.

Tabelle: Befehlsvorrat des AIM-65-BASIC

CLEAR	DEF FN
CONT	DIM
FRE(0)	END
LIST	FOR...(STEP)
LOAD	GOSUB
NEW	GOTO
PEEK	IF...GOTO
POKE	IF...THEN
RUN	LET (kann entf.)
SAVE	NEXT
DATA	ON...GOSUB
GET	ON...GOTO
INPUT	REM
POS	RESTORE
PRINT	RETURN
READ	STOP
SPC	USR
TAB	WAIT
ASC	ABS
CHR $	ATN
LEFT $	COS
LEN	EXP
MID $	INT
RIGHT $	LOG
STR $	RND
VAL	SIN
	SGN
	SQR
	TAN

Z-80-Einplatinen-Computer

„Nanocomputer" nennt SGS-Ates den Single-Board-Computer im *Bild*. Er arbeitet mit der 8-bit-CPU Z-80. Die im Doppel-Europa-Format gehaltene Platine enthält 4 KByte dynamisches RAM, das von der CPU selbst „aufgefrischt" wird, ein 2-KByte-Monitorprogramm und zwei Z-80-PIOs, d. h. 32 E/A-Leitungen. Die Ein- und Ausgabe geschieht über ein handliches, taschenrechnerähnliches, absetzbares Gerät im Hex-Code. Der Computer ist zu einem professionellen System (CLZ-80) ausbaubar. Ein Testbericht ist in Vorbereitung.



Derzeit eines der preiswertesten Z-80-Systeme: Der „Nanocomputer" von SGS-Ates mit dem dazugehörigen Hexadezimal-Eingabefeld

Einschränkungen muß man beim AIM-65 hinsichtlich der Korrekturmöglichkeiten hinnehmen. Wer vom PET 2001 gewohnt ist, ein Programm aufzulisten und dann mit dem Cursor einzelne Zeichen zu ändern oder gar einzufügen, muß sich umgewöhnen: Für die Korrektur einer Zeile muß diese komplett neu eingegeben werden; nur „Backspace" ist ein kleiner Trost für Leute, die sich hin und wieder vertippen – und wer tut das nicht...

Fehlermeldungen werden vom AIM-65 in einer abgekürzten Form ausgegeben; zum Glück allerdings nicht einfach als schlichter Zahlen-Code, sondern in Form zweier Buchstaben und dem Wort „ERROR". So bedeutet etwa SN ERROR soviel wie Syntax-Error, und OD ERROR heißt „Out of Data Error" – möge dieses recht selten vorkommen! Fe.

Textausgabe in Maschinensprache

Im FUNKSCHAU-Sonderheft HOBBYCOMPUTER 2 beschrieb E. H. Franke einen Weg, wie ein schwerwiegender Nachteil der Maschinensprache – auszugebende Texte müssen in einem besonders zu reservierenden Speicherbereich stehen – behoben werden kann.

```
PRTSTR  POP HL
        LD A,(HL)
        INC HL
        PUSH HL
        OR A
        RET Z
        RST 10H
        JR PRTSTR
```

Unterprogramm zur Textstring-Ausgabe. Der Text ist als ASCII-Folge direkt an den Aufruf dieses Unterprogrammes zu hängen und muß mit hex 00 (NOP) abgeschlossen werden

Sein 6502-Assemblerprogramm erlaubt es, den Text als ASCII-Bytefolge direkt hinter einen Unterprogrammaufruf zu schreiben, und das Programm wird hinter dem mit NOP abgeschlossenen Text fortgesetzt.

Die vorliegende Version für den TRS-80 unter Level-1-Basic funktioniert nach dem gleichen Prinzip. Die Z-80-Routine benötigt bemerkenswerterweise sogar weniger Befehle und Bytes als ihr 6502-Äquivalent.

Ing. (grad.) Rolf-Fr. Matthaei

Erzeugung von Pseudo-Rauschen mittels Software

Der integrierte Baustein MM 5837 erzeugt digitales Pseudo-Rauschen durch ein Schieberegister, in dessen erste Stufe das Ausgangssignal eines Exclusiv-Oder-Gatters eingegeben wird. Als Eingangssignal für dieses Gatter wird der Inhalt zweier Stufen des Schieberegisters verwendet. Diese Hardware-Lösung läßt sich auch durch Software mit einem Mikroprozessor verwirklichen.

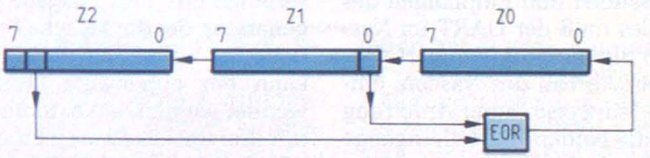


Bild 1. Schieberegister, gebildet aus den drei Zellen Z0, Z1 und Z2

Das Schieberegister besteht aus den drei Zellen Z0, Z1, Z2 (*Bild 1*). Als Eingangssignal für Z0 wird das Exclusiv-Oder, gebildet aus Bit 0 von Z1 und Bit 6 von Z2, verwendet. *Bild 2* ist das Programm für ein 6502-System (SYM). In diesem Programm ist zu beachten, daß bei den Schiebeoperationen das Carry-Bit mitverwendet wird. Zur Ausgabe wird Bit 0 von Tor A als Ausgang programmiert, so daß das digitale Pseudo-Rauschen dort abgenommen werden kann.

Noch ein Hinweis zur Programmier-Technik. Der Rücksprung wird durch die Befehlsfolge CLC, BCC M gebildet. Dies hat gegenüber des JMP ADRE Befehls den Vorteil, daß das Programm frei verschieblich (relocatable) im Speicher ist. Allerdings verlängert sich die Ausführungszeit um 2µs.

Beim Einsprung in das Programm wird angenommen, daß der Inhalt zumindest einer Zelle ungleich Null ist. Sollte dies nicht gewährleistet sein, so muß in eine Zelle ein beliebiger Zahlenwert geschrieben werden.

Dipl.-Ing. Ekkehard Flögel

```
                ZO:=0000
                Z1:=0001
                Z2:=0002

0200 A9 01      LDA=01          ; Bit 0 Tor A als Ausgang
0202 8D 03 A0   STA A003        ; programmieren
0205 A5 02      LDA 02          ; Bit 6 von Zelle Z2

0207 2A         ROL A           ; nach Bit 0 in den
0208 2A         ROL A           ; Akkumulator schieben
0209 2A         ROL A           ;
020A 45 01      EOR 01          ; Exclusiv-Oder mit Zelle Z1
020C 4A         LSR A           ; Bit 0 vom Akku ins CARRY-Bit
020D 26 00      ROL 00          ; Übernehmen.Die Zelleninhalte
020F 26 01      ROL 01          ; von ZO-Z2 einmal links
0211 26 02      ROL 02          ; schieben.
0213 A5 00      LDA 00          ; Inhalt (Bit 0) von ZO an
0215 8D 01 A0   STA A001        ; Tor A ausgeben.
0218 18         CLC             ;
0219 90 EA      BCC EA    0205  ; Rücksprung
```

Bild 2. 6502-Programm zum Erzeugen von Pseudo-Rauschen. Die Adressen A003 und A001 sind beim KIM-1 durch 1701 und 1700 zu ersetzen

Baudot-RTTY mit dem Nascom-1

Trotz aller Vorteile des ASCII-Codes ist doch die Verbreitung von Baudot im Amateurfunkfernschreiben (RTTY) – speziell auf den Kurzwellenbändern – noch überwältigend hoch. Nach dem ASCII-Betriebsprogramm für den Nascom aus Heft 4/1979 bringen wir deshalb hier ein gleichwertiges Programm für den Baudot-Code.

Für das Senden und Empfangen des Baudot-Codes muß der UART im Nascom-1 auf 5 Bits umprogrammiert werden. Wer bei Aufbau des Nascom entsprechende Hinweise in der Anleitung ignorierte, die beiden Einstelleingänge Cls 1 und Cls 2 aufzutrennen, muß jetzt unter Umständen den Sockel entfernen.

Bild 1 zeigt eine Schaltung, die das Programmieren des UART per Softwarebefehl gestattet. Der Inverter ist nötig, damit nach einem Reset wieder auf 8 Bits geschaltet wird. Nach dem Start des Baudot-Programms werden Cls 1 und Cls 2 auf Low gesetzt.

Selbstverständlich benötigt man zur seriellen Ein- und Ausgabe einen Taktgenerator, der die 16fache Frequenz der Baudrate aufweisen muß. Für 100 Bd kann der eingebaute Oszillator verwendet werden; ein externer Generator mit drei umschaltbaren Frequenzen für 45,5 Bd, 50 Bd und 75 Bd dürfte die beste Lösung darstellen (*Bild 2*).

Gegenüber dem in Heft 4/1979 vorgestellten ASCII-RTTY-Programm ergibt sich eine kleine Änderung in der Bedienung. Bei Empfang kann mit Hilfe des %-Zeichens (Shift 5) der Ziffern- bzw. Buchstaben-Modus geändert werden, wenn ein Umschaltzeichen verloren ging. Als Empfangs-Konverter geeignet ist z. B. der PLL-Konverter aus Heft 9/79 bzw. 16/79. Natürlich empfiehlt sich speziell für Kurzwelle die Verwendung brauchbarer Filter-Konverter. Für die Erzeugung der Töne für Mark und Space eignet sich jeder beliebige AFSK-Generator, so auch der in Heft 15/1978 beschriebene. *Bild 3* zeigt den vollständigen Hex-Dump.

Karl Schedler, DC2MP

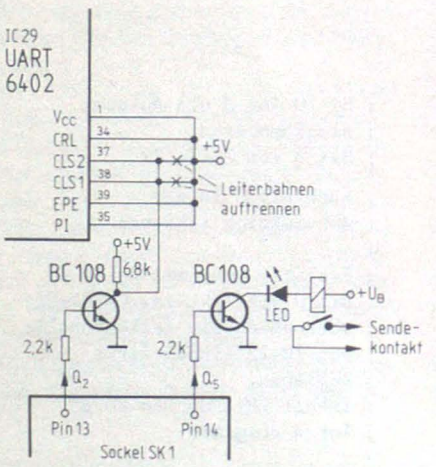
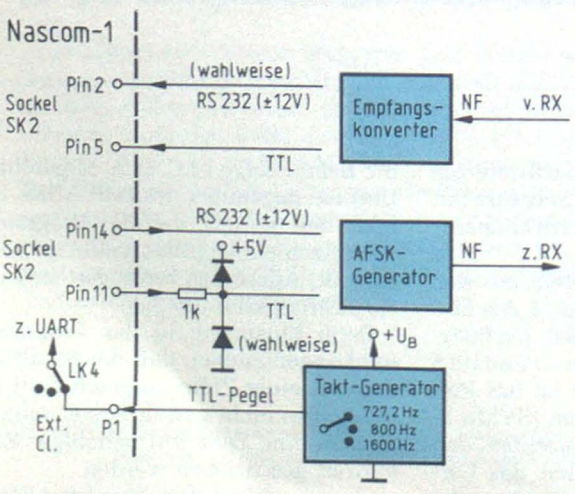


Bild 1. Hardware zur 5/8-bit-Umschaltung des UART per Softwarebefehl



◄ Bild 2. Anschluß von Taktgeber, Empfangskonverter und AFSK-Generator an den Nascom-1

```
0c50 c3 9d 3e 04 d3 00 32 00 0c 3e ff 32 51 0c ef 54
0c60 45 58 54 3f 1f 00 21 a0 0f 11 e0 0f cd 4d 0c 30
0c70 fb fe 40 28 19 fe 24 ca dc 0d cd 4a 0c fe 1d 20
0c80 03 2b 18 eb cd 87 0d 3a 50 0c fe 56 20 de 36 00
0c90 cd 40 02 2a 18 0c e5 21 d0 0b 22 18 0c ef 43 41
0ca0 4c 4c 20 20 20 52 54 54 59 20 20 00 21 e6 0b 22
0cb0 18 0c ef 46 49 46 4f 3a 20 20 00 3e 45 32 e0 0b
0cc0 e1 22 18 0c cd 4d 0c fe 23 28 93 fe 40 28 17 fe
0cd0 25 ca 92 0f db 02 17 30 eb db 01 cd 77 0f cd 57
0ce0 0d cd 4a 0c 18 de cd f1 0d 3e ff 32 51 0c cd 40
0cf0 02 21 e0 0b 36 53 21 00 0e 11 00 0e d9 21 a0 0f
0d00 7e 23 b7 28 07 d9 cd 6c 0d d9 18 f4 d9 06 ff 3a
0d10 50 0c fe 56 28 16 cd 35 00 cd 35 00 cd 4d 0c 30
0d20 0b fe 40 ca 3f 0d cd 4a 0c cd 6c 0d 10 e1 3a 50
0d30 0c fe 4c 28 06 db 02 87 fc a4 0d 06 01 18 d0 3a
0d40 50 0c fe 46 20 05 cd a0 0d 18 f4 db 02 cb 77 28
0d50 fa cd f1 0d c3 90 0c fe 08 20 02 3e 1d fe 0d 28
0d60 58 fe 0a 28 5c 47 d6 01 78 f0 af c9 fe 1e 20 02
0d70 3e 0c fe 1d 20 02 3e 08 fe 1f 20 07 3e 0d cd 40
0d80 0f 3e 0a cd 40 0f c9 f5 3a 50 0c fe 56 28 0f f1
0d90 77 2c 7d bb 28 04 3e 46 18 02 3e 56 18 14 f1 c9
0da0 db 02 87 f0 1a d3 01 1c 7b bd 20 04 3e 4c 18 02
0db0 3e 46 32 ed 0b 32 50 0c c9 cd d0 0d 21 8a 0b 18
0dc0 0b cd d0 0d e5 cd ea 0d cd d0 0d e1 22 18 0c c9
0dd0 2a 18 0c 7e fe 5f 20 02 36 20 af c9 21 a0 0f 7e
0de0 2c b7 ca 90 0c cd 4a 0c 18 f5 f5 c5 d5 e5 c3 95
0df0 01 3e 20 c3 53 00 05 25 f9 77 25 05 d7 bd 05 25
m
0f00 ff ff 45 33 0a 0a 41 2d 20 20 53 27 49 38 55 37
0f10 0d 0d 44 24 52 34 4a 07 4e 2c 46 21 43 3a 4b 28
0f20 54 35 5a 2b 4c 29 57 32 48 ff 59 36 50 30 51 31
0f30 4f 39 42 3f 47 26 01 01 4d 2e 58 2f 56 3d 00 00
0f40 cd 5c 0f e0 4f e6 01 e5 21 51 0c be 77 e1 28 07
0f50 cd 5c 0f 1f cd 87 0d 79 1f c3 87 0d e5 c5 21 00
0f60 0f 01 40 00 ed b1 e2 6e 0f 2b 7d c1 e1 c9 3a 51
0f70 0c e6 01 f6 00 18 f4 e5 21 51 0c 17 e6 3e b6 e5
0f80 21 00 0f 85 6f 7e e1 b7 28 04 fe 01 20 02 77 af
0f90 e1 c9 e5 21 51 0c af be 20 01 3c 77 e1 c3 d4 0c
0fa0 48 49 45 52 20 49 53 54 20 44 43 32 4d 50 20 4d
m
```

Bild 3. Hexadezimales Programmlisting. Mit ihm wird aus dem Nascom-1 ein komfortabler Baudot-Fernschreiber

Michael Klein

Festkomma-Addition und Subtraktion mit dem Z 80

In diesem Aufsatz werden Beispiele arithmetischer Programme angegeben. Probleme dieser Art kommen in der Praxis des öfteren vor. Gleitkommaarithmetik oder Multiplikationen von mehrstelligen BCD-Zahlen sollen hier nicht behandelt werden.

Hier soll kurz skizziert werden, wie man zwei 16stellige Dezimalzahlen mit dem Z-80-KIT addieren und subtrahieren kann. Dabei dürfen nur Zahlen gleichen Vorzeichens Verwendung finden. Die Routinen bieten keine Möglichkeit, ein Vorzeichen zu verarbeiten.

Es werden zwei Programme vorgestellt. Eines dient zur Addition zweier maximal 16stelliger Dezimalzahlen, das andere dient zur Subtraktion. Schon an dieser Stelle kann festgestellt werden, daß die Beschränkung auf 16stellige Zahlen völlig willkürlich ist. Genauso gut könnte man auch hundertstellige Zahlen mit einem nur wenig geänderten Programm verarbeiten.

Um mit Zahlen arbeiten zu können, muß man immer ein sog. „Format" vereinbaren, in dem man sie abspeichert. Bei uns soll eine Vereinbarung gelten, wie sie in *Bild 1* dargestellt ist. Wir verwenden aufeinanderfolgende Speicherzellen des RAM, um unsere Zahlen unterzubringen. Dabei stehen immer zwei Stellen der Zahl in einem Byte. Da wir es mit Festkommazahlen zu tun haben, braucht für das Komma selbst kein Speicherplatz vorgesehen zu werden. Ferner sehen wir: Die höherwertigen Stellen der Zahl, die verarbeitet werden soll, stehen auf höheren Adressen.

Für unser Programm wählen wir eine großzügige Speicheraufteilung, die es sogar ermöglicht, ohne Änderungen den Rechenbereich zu erweitern. Wir vereinbaren die in *Bild 2* gezeigte Aufteilung.

Die Zeiger HL, IX, DE verwenden wir in der gezeigten Weise, um Speicherstellen innerhalb eines 16-Byte-Bereiches anzusprechen.

Sehen wir uns nun das Flußdiagramm *Bild 3* an. Es zeigt, wie man ein Programm aufbauen kann, das zwei Zahlen, die im gezeigten Format abgespeichert sind, addiert und das Ergebnis in den Speicherzellen ab 3C90 ablegt.

Im ersten Operationsblock wird festgelegt, wieviele Bytes addiert werden. In unserem Falle sind es 8 Byte, also 16 Stellen. Anschließend werden die drei Zeiger HL, IX, DE gesetzt. Dann wird das Carryflag gelöscht, denn es könnte ja noch von einer vorhergehenden Operation her gesetzt sein und Rechenfehler verursachen. Im übrigen gibt es beim Z 80 keinen Befehl, um das Carryflag zu löschen. Man verfährt deshalb so, daß man mit dem Befehl SCF das Carryflag setzt und es anschließend mit dem Befehl CCF komplementiert, d. h. löscht. Der Befehl XOR A hätte die gleiche Wirkung.

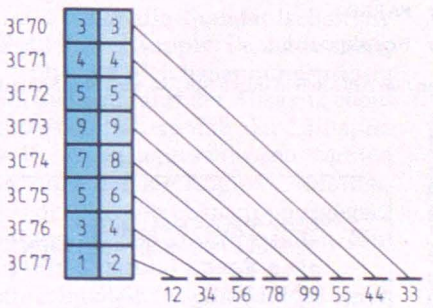


Bild 1. Abspeichern der Zahl 1234567899554433 im RAM; zwei Stellen finden jeweils in einem Byte Platz

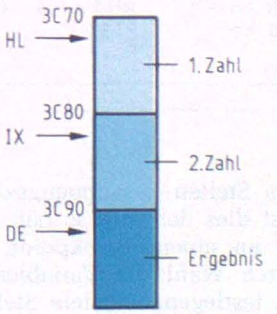


Bild 2. Aufteilung des Speichers für die beiden Summanden und das Ergebnis

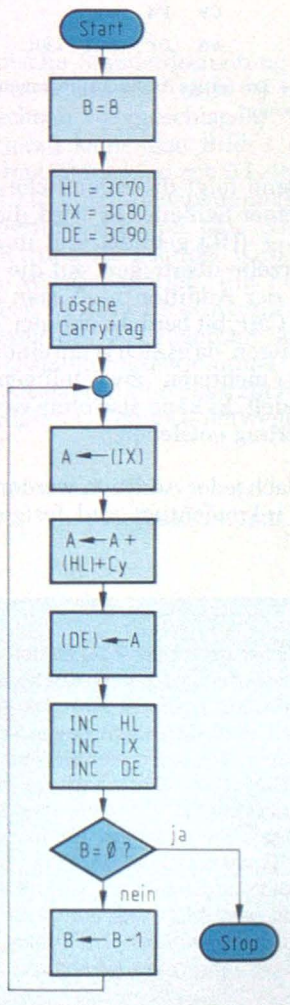


Bild 3. Flußdiagramm zur Addition zweier 16stelliger Zahlen

```
3C50  06  LD B,08          ; 8 Byte entsprechend 16 Stellen
  51  08                   ; können addiert werden
  52  21  LD HL,3C70       ; Zeiger für ersten Summanden
  53  70
  54  3C
  55  DD  LD IX,3C80       ; Zeiger für zweiten Summanden
  56  21
  57  80
  58  3C
  59  37  SCF              ; Setze Carryflag
  5A  3F  CCF              ; Complementiere Carryflag
  5B  11  LD DE,3C90       ; Ergebnis ab 3C90
  5C  90
  5D  3C
MARKE: 5E  DD  LD A,(IX+d)     ; Lade A mit einem Summanden
  5F  7E
  60  00
  61  8E  ADC A,(HL)       ; Addiere zweiten Summanden
  62  27  DAA              ; Dezimale Anpassung
  63  12  LD (DE),A        ; Lade (DE) mit Ergebnis
  64  23  INC HL           ; ERhöhe alle Zeiger um 1
  65  13  INC DE
  66  DD  INC IX
  67  23
  68  10  DJNZ MARKE      ; Wenn noch nicht fertig, gehe nach
  69  F4                   ; MARKE
  6A  DF  RST 18H          ; Breakpoint
```

Bild 4. Das fertige Additionsprogramm. Ersetzt man den ABC-Befehl durch SBC, so wird eine Subtraktion ausgeführt

Dann folgt die eigentliche Addition in einer Schleife. Es wird die Summe (IX) + (HL) gebildet und in die Speicherzelle übertragen, auf die DE zeigt. Bei der Addition muß man natürlich das Carrybit berücksichtigen, d. h. mitaddieren, da es sich ja um eine Addition von mehreren zweistelligen Zahlen handelt. Es kann also ohne weiteres ein Übertrag entstehen.

Nach jeder Addition werden alle Zeiger inkrementiert und festgestellt, ob bereits alle Stellen zusammengezählt wurden. Ist dies der Fall, so hält das Programm an einem Breakpoint an. Allein durch Wahl der Variablen B kann man festlegen, wieviele Stellen addiert werden sollen. Größter Wert von B ist OF. Will man mehr Stellen addieren, so muß man Hl, IX, DE anders setzen.

Das Additionsprogramm kann sehr leicht in ein Subtraktionsprogramm umgewandelt werden. Man ersetzt dazu lediglich den Befehl ADC A, (HL) durch den Befehl SBC A, (HL).

```
6502-PROGRAMM  FUER
32STELLIGE BCD-ADDITION
0200 F8      SED
0201 18      CLC
0202 A2 0F   LDX #0F
0204 B5 00   LDA 00,X
0206 75 10   ADC 10,X
0208 95 20   STA 20,X
020A CA      DEX
020B 10 F7   BPL 0204
020D 60      RTS
```

Sicher gehört der Z 80 zu den modernsten Mikroprozessoren des heutigen Angebotes. Trotzdem wird gerade an diesem Beispiel deutlich, daß er wegen seiner erzwungenen Kompatibilität mit dem „Großvater" 8080 manchmal etwas umständlicher zu programmieren ist als einige seiner Kollegen, wie ein Vergleich z. B. mit dem 6502 zeigt. Die „Endlosaddition" mit dem Z 80 benötigt 26 Bytes, mit dem 6502 aber nur 14 Bytes, also kaum mehr als die Hälfte. (Liebe Z-80-Fans, nichts für ungut...)

µC-Nachrichten

Die Münchener Firma Scantec liefert einen **Magnetblasenspeicher,** der sich für den Mikrocomputer SBC-80 bzw. für ein beliebiges Intel-Multibus-System eignet. Die Kapazität beträgt 92304 Byte und ist zur einfacheren Adressierung „memory-mapped" organisiert. Der Speicher heißt MBB-80 und enthält acht 92-Kbit-Magnetblasenmoduln von Texas Instruments.

□

Wem eine gewöhnliche Floppy-Disk für den TRS-80 nicht genügt, kann jetzt diesen Mikrocomputer mit **10-MByte-Magnetplatten** ausrüsten. MM-Computer (im bayerischen Prien am Chiemsee) liefert die dazu erforderliche Hardware und ein modifiziertes Platten-Betriebssystem für den TRS-80, das mit dem Standard-Betriebssystem kompatibel ist.

□

Von Neumüller ist ein **EPROM-Programmierzusatz** für den Mikrocomputer SYM-1 zu haben. Außer der kleinen Platine, die direkt an den SYM gesteckt werden kann, ist nur noch ein 256 Byte umfassendes Programm erforderlich, dessen Listing mitgeliefert wird. Das „SYMP" eignet sich für die Chips 2758, 2716, 2516, 2532 und 2732.

□

Einen **Magnetblasenspeicher mit 1 MByte Kapazität** entwickelten Wissenschaftler bei Bell – genug Speicherplatz, um ein Telefonbuch mit 35 000 Nummern unterzubringen. Dabei wird eine Ionen-Implantationstechnik verwendet, die auf einer Fläche von nur 1,3 Quadratzoll 8,7 Mio. Bits speichern kann; pro Bit werden 64 µm² benötigt. Die maximale Speicherkapazität wäre – bei Verzicht auf Redundanz – sogar 11,5 Mio Bits.

□

National Semiconductor lieferte kürzlich das millionste elektronische **Basketball-Spiel** an Mattel. Charles Sporck übergab das vergoldete Gerät in feierlichem Rahmen Mattel's Präsident Ray Wagner.

□

Drei neue Taschenrechner stellte Hewlett-Packard in den USA vor. Es handelt sich um die Typen HP-34 C (20 Datenregister oder 210 Programmzeilen, Integrationsfunktionen), HP-33C (Nachfolger des HP-33 mit CMOS-Speicher) und HP-38 C (Nachfolger des HP-38 E mit CMOS-Speicher). In den USA kosten die Rechner 120...150 Dollar.

Ing. (grad.) Uwe Kornnagel

Systemgerechte Programmierung des PET

Unter systemgerechter Programmierung versteht man, ein Programm so abzufassen, daß Speicherbedarf und Rechenzeit möglichst klein gehalten werden. Dieser Aufsatz geht sowohl auf Basic- als auch auf Maschinenprogrammierung ein. Die Adressen gelten für den PET 2001; beim CBM 3001 wurden sie leider geändert.

Ein Byte von der Tastatur holen

Ein sehr häufiges Problem ist das Einlesen eines Bytes über die Tastatur. Es läßt sich zwar mit einer Schleife lösen, aber man verbraucht unnötigen Speicherplatz. Statt:
10 GET T$:IF T$ = " " THEN 10,
schreibt man besser:
10 WAIT 525, 1 : GET T$

Daten auf Kassette schreiben

Oft gibt es Schwierigkeiten beim Schreiben von Daten auf eine Kassette. Commodore zeigt im Handbuch zwar eine Lösung auf, sie ist aber viel zu speicheraufwendig. Man sollte besser folgendes Unterprogramm schreiben und es nach jedem PRINT # n aufrufen. Das hat den Vorteil, daß man mehrere PRINT # n mit diesem Programm kontrollieren kann und noch weniger Speicherplatz benötigt.
10000 REM TAPE CONTROL
10010 BU% = PEEK(625) : IF BU% > BV% THEN 10040
10020 POKE 59411,53 : TT = TI
10030 IF TT – TI > – 40 THEN 10030
10040 POKE 59411,61 : BV% = BU% : RETURN
Für das Laufwerk 2 gilt die obige Routine mit folgenden Änderungen: Bei dem PEEK-Befehl 626 statt 625; bei den POKE-Befehlen POKE 59456,239 statt POKE 59411,53 und POKE 59456,255 statt POKE 59411,61.

Die USR-Funktion

USR, bei anderen Maschinen auch als USER bekannt, erlaubt ein Unterprogramm in Maschinensprache aufzurufen oder Variable auszutauschen. Die Startadresse des Maschinenprogrammes muß vorher erst mit POKE-Befehlen in die Adressen 1 und 2 geschrieben werden. Beispiel: Ein Programm beginnt ab Adresse 826 = 03 3A.
10 POKE 1,58 : POKE 2,3. 58 = 3A 3 = 03

Das höherwertige Byte steht in Adresse 2, das niederwertige in Adresse 1. Der Syntax lautet dann B = USR(A) oder PRINT USR(A). Die Variable A steht dann im Fließkomma-Akku (Adresse B0...B5). Um sie in Festkomma-Format umzuwandeln, muß der erste Befehl im Maschinenprogramm ein JSR D0A7 – Befehl (20 A 7 D0) sein. Das höherwertige Byte der Festkommazahl steht dann in der Adresse B3, das niederwertige in Adresse B4. Zur Ausgabe vom Maschinenprogramm muß die Festkommazahl

```
1 DEFFNY(X)=:::::::::::::::::::::::::::::::
2 IF JP=0 THEN 10000
5 REM PROGRAMM EINGEBEN
10000REM STRING-BASIC-COMPILER
10010 DATA 169,58,162,50,157,11,4
10020 DATA202,208,250,96,96
10030 IF PEEK(836) = 96 THEN 10060
10040 FOR I=826 TO 836
10050 READ FC%:POKEI,FC%:NEXT
10060 IF PEEK(1036)<>58 THEN JP = 1:GOTO 1
10080 INPUT"Y=";FC$:KL=1036
10090 FOR I=1 TO LEN(FC$)
10100 FC%=ASC(MID$(FC$,I,1))
10110 IF MID$(FC$,I,1)="+" THEN FC%=170
10120 IF MID$(FC$,I,1)="-" THEN FC%=171
10130 IF MID$(FC$,I,1)="*" THEN FC%=172
10140 IF MID$(FC$,I,1)="/" THEN FC%=173
10150 IF MID$(FC$,I,1)="I" THEN FC%=174
10160 IF FC%>169 AND FC%<176 THEN 10300
10170 IF MID$(FC$,I,3)="SGN" THEN FC%=180
10180 IF MID$(FC$,I,3)="INT" THEN FC%=181
10190 IF MID$(FC$,I,3)="ABS" THEN FC%=182
10200 IF MID$(FC$,I,3)="USR" THEN FC%=183
10210 IF MID$(FC$,I,3)="SQR" THEN FC%=186
10220 IF MID$(FC$,I,3)="RND" THEN FC%=187
10230 IF MID$(FC$,I,3)="LOG" THEN FC%=188
10240 IF MID$(FC$,I,3)="EXP" THEN FC%=189
10250 IF MID$(FC$,I,3)="COS" THEN FC%=190
10260 IF MID$(FC$,I,3)="SIN" THEN FC%=191
10270 IF MID$(FC$,I,3)="TAN" THEN FC%=192
10280 IF MID$(FC$,I,3)="ATN" THEN FC%=193
10290 I=I+2
10300 POKE KL,FC%:KL=KL+1 :  NEXT : RUN
```

Mit diesem Programm ist es möglich, eine Funktion bei laufendem Programm mittels des INPUT-Befehles zu definieren. Sie kann dann mit FNY(X) aufgerufen werden. Das Anwenderprogramm steht zwischen den Zeilen 5 und 10 000

wieder in eine Fließkommazahl umgewandelt werden. Die Umwandlung geschieht mit dem Programm auf Adresse D278. Der letzte Befehl des Maschinenprogrammes muß demnach der Befehl JMP D278 (4C 78 D2) sein. Der SYS-Befehl ist ebenfalls ein Aufruf für ein Maschinenprogramm, hierbei können aber keine Variablen direkt übergeben werden.

Formeleingabe per INPUT

Für den Programmierer stellt sich oft die Aufgabe, eine Funktion über einen INPUT-Befehl einzugeben, und diese dann in das Programm zu inplementieren. Bei den meisten Maschinen ist es fast unmöglich, beim PET läßt es sich aber mit relativ wenig Aufwand realisieren. Sie müssen nur einige Regeln beachten: Die Funktion muß immer in der ersten Zeile stehen, und es muß genug Platz für die Funktion freigehalten werden.

Das Programm im *Bild* ist ein Beispiel, wie man eine Funktion in Basic inplementiert. In die Zeile „1 DEFFNY(X)=" sind 50 Doppelpunkte zu setzen, damit ist reichlich Speicherplatz für die Funktion reserviert. Der Befehl SYS(826) löscht die Funktion.

Bei „,", „;" oder „:" in der Formel muß der INPUT mit " begonnen werden.

I/O-Programmierung

Es ist oft wünschenswert, Ein- oder Ausgabe mit Maschinenprogrammen zu verknüpfen. Für diese Anwendungen stehen einige Systemroutinen zur Verfügung:

Ein Zeichen aus dem Akku drucken (Bildschirm): FFD2
INPUT ein Zeichen in den Akku: FFCF
GET ein Zeichen in den Akku: FFE4
Der PET bietet ebenfalls die Möglichkeit, ein Textfeld im Maschinenprogramm auszudrucken. Das letzte Byte im Textfeld muß 00 sein. Die Anfangsadresse des Textfeldes steht in den Prozessorregistern A und Y; das höherwertige Byte im Y-, das niederwertige Byte im A-Register. Die Ausgabe erfolgt dann mit JSR CA27 (20 27 CA).
Will man einen bestimmten Speicherbereich auf Kassette schreiben, ist dies auch ohne TIM-Monitor möglich. Die Start- und Endadressen setzt man mit POKE-Befehlen.

Das niederwertige Byte der Startadresse „poket" man in 247, das höherwertige in 248. Die Endadresse schreibt man in 229 und 230. Das Schreiben auf Kassette geschieht dann mit dem Aufruf SYS(63153), allerdings ohne File-Namen. Soll die Aufzeichnung File mit einem Namen versehen werden, so muß die Länge des Namens in Adresse 238 und dessen Anfangsadresse in 249,250 geschrieben werden.

Will man dieses Programm wieder laden, so muß man genau wie beim Schreiben die Startadresse in die Adressen 247 und 248 „poken" und dann die Laderoutine mit SYS(62306) aufrufen.

Meine Meinung...

Brauchen wir PASCAL?

Man braucht kein Branchenkenner zu sein, um zu wissen, daß die höhere Programmiersprache PASCAL allgemein als „die Sprache der Zukunft" angesehen wird. Dieser Trend, so berechtigt er einerseits sein mag, ist dabei, in einer regelrechten Hysterie auszuarten. Die Anzeichen dafür sind deutlich:

Kaum einer kennt mehr von der Sprache als die Schlagworte, die in aller Munde sind. Trotzdem – oder besser gesagt gerade deshalb – will sie jeder so schnell wie möglich in sein System packen und damit arbeiten.

Dieser Wunsch ist mit Sicherheit ein Schritt vorwärts, wenn er von einem Profi aus der Industrie geäußert wird, der sehr große Programme zu erstellen und zu warten hat. Der Bastler, der sich mit dem Gedanken trägt, PASCAL für sich einzusetzen, sollte vorher einige Dinge darüber erfahren, damit es für ihn nicht zu einem Bumerang wird.

PASCAL wurde ursprünglich an der Eidgenössischen Technischen Hochschule Zürich mit dem Ziel entwickelt, eine leicht lehrbare Sprache zur Verfügung zu haben. Überraschenderweise entstand ein Werkzeug, das nicht nur zum disziplinierten Programmieren zwang, sondern nebenbei auch noch recht leistungsfähig war. Hier muß bereits die erste Anmerkung gemacht werden: Das Attribut „leicht lehrbar" und damit „erlernbar" muß im Vergleich zu einer Sprache wie FORTRAN gesehen werden. Es gilt ganz bestimmt nicht für Hobbyprogrammierer, sondern für Leute mit einigen Semestern Informatikstudium. Vergleicht man den Schwierigkeitsgrad von BASIC und PASCAL, dann ist letztere eine schwer erlernbare Sprache.

Der große Fortschritt, der mit PASCAL erreicht wird, liegt darin, daß sehr lange Programme automatisch übersichtlich, leicht zu ändern und leicht lesbar werden. Dinge, die in der Industrie viel Geld sparen können. Der Preis, der dafür zu zahlen ist, besteht z. B. darin, daß man lange überlegen muß, bevor man ein PASCAL-Programm tatsächlich niederschreiben kann. Der Lösungsweg muß in allen Einzelheiten durchdacht sein, ehe man mit der sogenannten Codierung beginnt. Nur bei langen Programmen wird deshalb die gesamte Entwicklungszeit kürzer sein als bei anderen Sprachen.

Wieder ergibt sich für den durchschnittlichen Hobbyprogrammierer kein Vorteil gegenüber BASIC. Im Gegenteil: er wird kaum jemals so lange Programme schreiben, daß er in PASCAL schneller ist, und er ist in der Regel Programmentwickler und -benutzer in einer Person – das Problem der Lesbarkeit besteht für ihn also nicht in dem Maße wie für die Industrie. Überdies muß man bedenken, daß all die schönen Vorteile von PASCAL natürlich auch mit anderen Sprachen erreicht werden können – die einzige Voraussetzung dafür ist die Disziplin des Programmierers. Auch in BASIC lassen sich übersichtliche, modular aufgebaute, gut dokumentierte Programme schreiben.

Zu guter Letzt darf man eines nicht vergessen: PASCAL-Compiler brauchen wesentlich mehr Speicherplatz als BASIC-Interpreter. Um auf dem TRS-80 PASCAL laufen zu lassen, braucht man beispielsweise 48 KByte RAM und zwei Disketten-Laufwerke. Dem Interessenten stehen also nicht unerhebliche Kosten ins Haus.

Dies alles sollten Sie bedenken, wenn die Industrie Sie demnächst davon überzeugen will, daß auch Sie PASCAL brauchen!

Rudolf Hofer

Hans-Georg Joepgen

Vorsicht - Falle!

Die „Null-Probleme" binärer Basic-Interpreter

Wenn ein Programm auch nach gründlicher Überprüfung einen Mikrocomputer noch immer zu merkwürdigem Verhalten veranlaßt, dann kommt irgendwann der Zeitpunkt, an dem der Programmierer an seiner Maschine zu zweifeln beginnt und sich einem Hardwarefehler oder einer System-Unvollkommenheit auf der Spur glaubt. Zu Unrecht in den meisten Fällen: Die Erfahrung zeigt, daß dann zumeist doch noch immer menschliche Unvollkommenheit und nicht der vermutete Maschinenfehler im Spiele ist. Manchmal jedoch ist wirklich der Computer schuld.

Es ging um die Entwicklung eines recht umfangreichen Programms, mit dessen Hilfe dem PET etwas beigebracht werden sollte, was sonst nur wesentlich größere Computer auf Anhieb beherrschen: Operationen aus der höheren Mathematik an komplexen Zahlen [1]. Dabei zeigte ein Unterprogramm ein höchst seltsames Benehmen. Es sollte, um Divisionen durch Null zu verhindern, immer dann eine sehr kleine Ersatzzahl nahe bei Rechner-Null in eine Variable einlesen, wenn sich in der Variablen eine absolute Null befand. Kein unbilliges Verlangen, aber der Rechner weigerte sich und dachte offenbar überhaupt nicht daran, bei N=0 den gewünschten vorzeitigen Rücksprung auszuführen. Aufgrund früherer Erfahrungen zögerte der Programmierer lange, den Fehler beim Rechner zu suchen. Als die fraglichen Operationen dann jedoch, von allem Beiwerk befreit und völlig isoliert, noch immer zum gleichen Ergebnis führten, da wurde klar: Hier ging's um einen jener seltenen Fälle, in denen der Computer unrecht hat, nicht der Mensch [2].

Zur Erläuterung diene *Bild 1*. In Zeile 100 wird ein Datenkanal mit der Ordnungsnummer 1 geschlossen und sogleich wieder geöffnet, wobei der Kanal dem Bus-Teilnehmer mit der Adresse 4 – hier einem Drucker 9871 A von Hewlett-Packard – mit der Betriebsart „nur Empfangen" zugewiesen wird. Die Zeilen 110 bis 130 bilden eine Schleife, die anfänglich die Variable N mit dem Wert minus 1 lädt und sie sodann um jeweils 0,1 erhöht, bis 1 erreicht ist. Zur Diagnose wird in Zeile 120 der jeweils aktuelle N-Wert auf den Datenkanal 1 geschaltet und damit ausgedruckt.

Zu erwarten war, daß im elften Durchgang der Wert von N Null erreicht haben müsse, denn minus 1 plus 1 ergibt nun einmal Null. Nicht so jedoch in diesem Fall für den PET: Er kommt auf rund 2 mal 10^{-10}, wie *Bild 2* zeigt; eine Abweichung, die in den meisten Fällen nicht stört, denn der Fehler ist ja nur der hundertste Teil vom Promille eines Promille, aber dennoch peinlich, wenn man in diesem Augenblick auf Null testet und etwa IF N = 0 THEN ... verlangt.

Schizophrenie, Bewußtseins-Spaltung, eine spezielle PET-Krankheit – oder etwa ein Krankheitsbild, für das auch andere Mikrocomputer anfällig sind? Um diese Frage zu klären, schrieb der Verfasser ein kleines Testprogramm

```
100 CLOSE1:OPEN1,4,1
110 FORN=-1TO1STEP.1
120 PRINT#1,N
130 NEXT
READY.
```

Bild 1. Hier ein Programmsegment, das den PET zwingt, sein merkwürdiges „Nullverständnis" zu offenbaren: Eine Schleife, die den Rechner von –1 bis +1 in Schritten zu 0,1 zählen läßt und ihn veranlaßt, den jeweiligen Zählerwert auszudrucken

```
]
7  REM FALLENTEST
8  REM ----------
9  REM
10  FOR N =  - 1 TO 1 STEP .1
15  PRINT N
20  IF N = 0 THEN  PRINT : PRINT
     "HIER ZEILE 20, N EXAKT GLEI
     CH NULL. ": PRINT
30  NEXT : END
```

Bild 3. Bewußtseins-Spaltung in Sachen Null: Eine typische PET-Krankheit, oder eine Schwäche auch anderer Mikrocomputer? Dieses Mini-Programm für den Euro-Apple soll's verraten: In Zeile 20 wird der Wert von N auf Null getestet

```
-1
-.9
-.8
-.7
-.6
-.5
-.4
-.3
-.2
-.0999999997
 2.91038305E-10
 .1
 .2
 .3
 .4
 .5
 .6
 .7
 .800000001
 .900000001
```

◀ Bild 2.
Das kommt dabei heraus, wenn der PET die Testroutine aus Bild 1 durchläuft: Bis –0,2 verhält sich der Rechner erwartungsgemäß, dann tauchen erste Ungenauigkeiten auf. Dort, wo eigentlich der Wert 0 erscheinen müßte, wird's kritisch: Der PET kommt nicht bei 0 vorbei

Bild 4. ▶
Der Euro-Apple läuft nicht nur in die gleiche Falle wie der PET, die Abweichungen von den erwarteten Werten sind darüber hinaus haargenau die gleichen wie auf der anderen Maschine. Dies läßt den Verdacht entstehen, daß hier nicht schlichte Maschinenfehler, sondern prinzipielle Schwächen binärer Basic-Interpreter am Werke sind

```
-1
-.9
-.8
-.7
-.6
-.5
-.4
-.3
-.2
-.0999999997
2.91038305E-10
.1
.2
.3
.4
.5
.6
.7
.800000001
.900000001
```

```
]
7  REM FALLENTEST
8  REM ----------
9  REM
10  FOR N =  - 1 TO 1 STEP .1
12  GOSUB 100
15  PRINT N
20  IF N = 0 THEN  PRINT : PRINT
     "HIER ZEILE 20, N EXAKT GLEI
     CH NULL.": PRINT
30  NEXT : END
96  REM
97  REM ABHILFE
98  REM -------
99  REM
100 N =  INT (N * 1E5 + .5) / 1E5
     : RETURN
```

Bild 5. Abhilfe durch das Unterprogramm „Abhilfe": In Zeile 100 rundet der Rechner den Wert des Schleifenzählers N auf fünf Stellen nach dem Komma. Ein taugliches Gegenmittel gegen Rechner-Schizophrenie? Der Ausdruck wird's weisen

für den 2020 von ITT Pforzheim (*Bild 3*). Es verlangt vom Rechner das Gleiche, woran der PET scheiterte, mit der Ausnahme allerdings, daß hier zur Verdeutlichung der Test von N auf Null ausdrücklich eingebaut wurde. Und siehe da, wie *Bild 4* zeigt: Auch für den Euro-Apple ITT 2020 gilt, daß er auf dem Weg von –1 nach +1 niemals bei Null vorbeikommt. Und, höchst erstaunlich: Seine Pseudo-Null entspricht haargenau der des PET, auch die übrigen Abweichungen von den erwarteten Werten sind identisch.

Dem Fehler auf der Spur

Woran liegt's? Die anfängliche Vermutung, es könne sich hier ein Interpreter-Fehler zeigen, bestätigte sich nicht, denn auch andere Binär-Basic-Interpreter zeigen ein gespaltenes Null-Bewußtsein. Die Ursache ist darin zu suchen, wie Binär-Basic-Interpreter Zahlen abbilden: Durch eine Mantisse, der ein Exponent folgt. Und für all das steht nur eine begrenzte Stellenzahl zur Verfügung. Die Abbildung dezimaler Zahlen durch ein derartiges duales Symbol bringt es nun aber mit sich, daß der Wert von beispielsweise dezimal –0,1 eben nicht nur aus vier Zeichen besteht wie im Dezimalsystem, sondern aus einer wesentlich größeren Gruppe Bits, als der Rechner für die Speicherung von Variablen bereitstellen kann. Den nicht mehr zu bewältigenden Rest muß er weglassen, was dann bei der Rückverwandlung ins dezimale Zahlen-Abbild in gewissen, glücklicherweise seltenen Fällen zu Erscheinungen wie der geschilderten führt.

Für mathematisch Interessierte in diesem Zusammenhang der Hinweis, daß es eine ganze Menge von endlichen Dezimalzahlen gibt, die sich in binärer Darstellung als Größen mit unendlich vielen Stellen erweisen – man sagt in der Fachsprache der Mathematik dazu, die Größen seien im betreffenden Zahlensystem inkommensurabel, nicht meßbar. Mit anderen Worten: Wir stehen hier nicht etwa vor dem Geiz von Mikrocomputer-Entwicklern, die ihren Rechnern zu wenig Speicherplatz für die Variablenaufbewahrung mit auf den Weg geben, sondern vor einem Naturgesetz.

Was ist zu tun?

Aufmerksamen Lesern wird aufgefallen sein, daß wir bis jetzt allein von „binären" Basic-Interpretern sprachen, und das geschah mit gutem Grund. Es gibt nämlich auch Mikrocomputer, die ihre Rechenarbeit nicht an binär verschlüsselten Größen verrichten, sondern Dezimalzahlen in den BCD-Code umsetzen [3]. Rechner von Hewlett-Packard gehören zum Beispiel dazu.

Dies hat mancherlei Nachteile: Man braucht mehr Speicherraum, die Rechengeschwindigkeit ist geringer, in vielen Fällen – bei gegebenem Variablenformat – auch die Genauigkeit kleiner, als wenn man rein binär verführe. Aber: Das Problem der Inkommensurabilität zwischen dezimal und binär abgebildeten Größen tritt in Rechnern mit BCD-Basic-Interpretern nicht auf und somit auch nicht die ominöse Falle mit der schizophrenen Null. Das hat eine Gruppe von Hobby-Computer-Freunden in Kalifornien dazu gebracht, sich vor einigen Monaten daran zu machen, einen BCD-Basic-Interpreter für den Apple zu schreiben. Man werde, wenn alles gut gehe, etwa Mitte 1980 mit der Arbeit fertig sein, erklärten die Interpreter-Künstler unlängst einem amerikanischen Reporter.

Schnelle Hilfe auf diesem Weg ist also nicht zu erwarten. Es bietet sich ein anderer Ausweg an – außer der trivialunsportlichen Möglichkeit selbstverständlich, schizophrenie-verdächtigen Segmenten beim Programmieren von Anfang an bewußt aus dem Weg zu gehen: Der Ausweg nämlich, beim Testen auf Null auf einen Teil der dem Rechner von Haus aus gegebenen Rechengenauigkeit zu verzichten und gewaltsam zu runden. Verzicht auf generelle Genauigkeit bringt hier, so paradox das klingen mag, ein Mehr an Genauigkeit im Sonderfall.

```
-1
-.9
-.8
-.7
-.6
-.5
-.4
-.3
-.2
-.1
0
HIER ZEILE 20, N EXAKT GLEICH NULL.
.1
.2
.3
.4
.5
.6
.7
.8
.9
1
```

Bild 6. Jetzt ist die Welt wieder in Ordnung. Nach dem Einfügen der Rundungsroutine „Abhilfe" benehmen sich Rechner wie PET und Euro-Apple endlich so, wie man's von einem anständigen Mikrocomputer erwartet: Die Zählerwerte stimmen, und auch der Test von N auf Null führt zu einem präzisen Ergebnis: Zeile 20 meldet sich und teilt mit, daß Null jetzt tatsächlich Null sei.

Bild 5 zeigt, um was es geht: In das Programm nach Ausdruck 3 ist in Zeile 12 die Anweisung zur Abarbeitung der Subroutine 100 eingefügt, die den sinnreichen Namen „Abhilfe" führt. Sie veranlaßt den Rechner, den Wert der Variablen auf fünf Stellen nach dem Komma zu runden. Eine Operation, die in den meisten Anwendungsfällen überhaupt nicht stört und die doch dafür sorgt, daß Null fortan Null ist, nicht mehr und nicht weniger; ein Unterprogramm, welches zuverlässig verhindert, daß Programmierer in die „Null-Falle" binärer Basic-Interpreter stolpern: *Bild 6* verrät, daß nach dem Einsatz von „Abhilfe" die Welt wieder in Ordnung ist – jetzt kommt der Rechner auf seinem Marsch über den Zahlenstrahl tatsächlich an Null vorbei.

Literatur

[1] *Joepgen, Hans-Georg:* Komfort-Programm zum Rechnen mit komplexen Zahlen. ELEKTRONIK 1979, Heft 17.

[2] *Handle, Dr. Franz G.:* Zahlendarstellung im PET. Sonderheft „HOBBY-COMPUTER 2", Franzis-Verlag, Oktober 1979.

[3] *Konz, M.:* 6502-Rechenroutinen. FUNKSCHAU 1979, Heft 19.

Dipl.-Ing. Willy Kunz

Eins aus sechzig

„1 aus 60" ist die Computerversion eines bekannten Zahlentricks. Der Mitspieler merkt sich eine Zahl zwischen eins und sechzig. Der Computer versucht, diese Zahl herauszufinden, indem er mehrere Male eine Auswahl von Zahlen zeigt und fragt, ob sich die gesuchte darunter befindet. Danach meldet er das richtige Resultat – sofern sein menschlicher Gegenspieler nicht gemogelt hat.

Das Programm wurde in CBASIC auf einem S-100-Hobbycomputer geschrieben. Die zwei wesentlichsten Unterschiede zu anderen BASIC-Übersetzern sind: 1. Unnötige, d. h. nicht angesprochene Zeilennummern dürfen weggelassen werden, 2. Variablennamen sind nicht auf einem Buchstaben beschränkt, sondern können bis zu 31 Zeichen umfassen. Das Programm läßt sich mühelos auf andere BASICs übertragen, wenn man entsprechende Zeilennummern einsetzt und die Variablennamen auf ein Zeichen verkürzt, z. B. CODE auf C, ANTWORT$ auf A$. Sollte ihr BASIC Strings nicht verarbeiten können, läßt sich die übliche Ausweichmethode anwenden, Eingaben von „NEIN" und „JA" durch Zahleneingaben zu ersetzen. Einer Erläuterung bedarf die Konstruktion

INPUT „text"; LINE A$

CBASIC druckt dabei zuerst den „text" aus und wartet dann auf eine Eingabe des Benutzers. Diese wird der Stringvariablen A$ zugewiesen. „LINE" bewirkt, daß die Eingabe nicht auf Einhaltung von Syntaxregeln geprüft wird; insbesondere darf A$ auch leer sein, d. h. nur aus einem „RETURN" bestehen. Sollte dies bei Ihrem BASIC nicht zulässig sein, müssen Sie darauf achten, daß Sie in Zeile 22 vor dem „RETURN" mindestens ein Zeichen eingeben.

CBASIC läßt mehrere Anweisungen auf derselben Zeile zu, wenn sie durch Doppelpunkte getrennt werden. Die meisten modernen BASIC-Übersetzer bieten diese Möglichkeit ebenfalls.

Der Ausdruck PRINT CHR$ (26) sendet das ASCII-Zeichen mit dem Dezimalwert 26 zum Terminal. Dies hat zur Folge, daß der Bildschirm gelöscht wird (hier ein ADM-3A von Lear Siegler). Eine ähnliche Wirkung hat die universellere Methode

```
FOR L = 1 TO 24
PRINT
NEXT L
```

Sie schiebt den alten Text aus dem Bildschirm heraus. Bei Verwendung

```
100 RESTORE
    PRINT CHR$(26)      REM  LOESCHT BILDSCHIRM DES ADM-3A
    PRINT TAB(30); "EINS-AUS-SECHZIG"
    PRINT TAB(30); "----------------"
    PRINT : PRINT : PRINT
X = 0                   REM  LOESUNGSZAHL INITIALISIEREN
PRINT TAB(15);
PRINT "MERKEN SIE SICH EINE GANZE ZAHL ZWISCHEN 1 UND 60."
PRINT TAB(15);
INPUT "DRUECKEN SIE DANN DIE TASTE 'RETURN'."; LINE LEER$
    FOR N = 1 TO 6
       GOSUB 1000      REM  ZEIGE ZAHLEN
       GOSUB 1100      REM  HOLE ANTWORT
       GOSUB 1200      REM  WERTE ANTWORT AUS
    NEXT N
PRINT : PRINT
PRINT "ICH WUERDE SAGEN, SIE HABEN SICH DIE ZAHL"; X; "GEMERKT."
PRINT
INPUT "NOCH EINMAL ?"; LINE ANTWORT$
IF LEFT$(ANTWORT$,1) = "J" THEN 100
STOP
REM ========================================================

REM    SUBROUTINEN BEGINNEN HIER
REM    ------------------------
REM    1. NAECHSTE 30 ZAHLEN DARSTELLEN

1000   PRINT CHR$(26)          REM  BILDSCHIRM LOESCHEN
       PRINT : PRINT : PRINT
       FOR I = 1 TO 6
           FOR K = 1 TO 5
              READ Z
              PRINT TAB(K*12);Z;
           NEXT K
           PRINT : PRINT
       NEXT I
       RETURN
REM    2. ANTWORT VOM SPIELER HEREINHOLEN
1100   PRINT
       INPUT "SEHEN SIE HIER IHRE ZAHL ?"; LINE ANTWORT$
       IF LEFT$(ANTWORT$,1) = "J" THEN 1105
       IF LEFT$(ANTWORT$,1) (()) "N" THEN 1100
1105   RETURN
REM    3. SCHLUESSELZAHL LESEN UND VERARBEITEN
1200   READ CODE
       IF LEFT$(ANTWORT$,1) (()) "J" THEN 1205
       X = X + CODE
1205   RETURN
REM    ZAHLENFELDER UND SCHLUESSELZAHLEN
DATA   1,3,5,7,9,11,13,15,17,19,21,23,25,27,29
DATA   31,33,35,37,39,41,43,45,47,49,51,53,55,57,59,1
DATA   2,3,6,7,10,11,14,15,18,19,22,23,26,27,30
DATA   31,34,35,38,39,42,43,46,47,50,51,54,55,58,59,2
DATA   4,5,6,7,12,13,14,15,20,21,22,23,28,29,30
DATA   31,36,37,38,39,44,45,46,47,52,53,54,55,60,13,4
DATA   8,9,10,11,12,13,14,15,24,25,26,27,28,29,30
DATA   31,40,41,42,43,44,45,46,47,56,57,58,59,60,13,8
DATA   16,17,18,19,20,21,22,23,24,25,26,27,28,29,30
DATA   31,48,49,50,51,52,53,54,55,56,57,58,59,60,31,16
DATA   32,33,34,35,36,37,38,39,40,41,42,43,44,45,46
DATA   47,48,49,50,51,52,53,54,55,56,57,58,59,60,46,32
END
```

Bild 1. BASIC-Programmauflistung für das Spiel „1 aus 60"

```
                    EINS-AUS-SECHZIG
                    ----------------

          MERKEN SIE SICH EINE GANZE ZAHL ZWISCHEN 1 UND 60.
          DRUECKEN SIE DANN DIE TASTE 'RETURN'.

          1          3          5          7          9
          11         13         15         17         19
          21         23         25         27         29
          31         33         35         37         39
          41         43         45         47         49
          51         53         55         57         59
SEHEN SIE HIER IHRE ZAHL ? JA
          2          3          6          7          10
          11         14         15         18         19


          42         43         46         47         50
          51         54         55         58         59
SEHEN SIE HIER IHRE ZAHL ? JA


          22         23         26         27         30
          31         34         35         38         39



          8          9          10         11         12
          13         14         15         24         25
          26         27         28         29         30
          31         40         41         42         43
          44         45         46         47         56
          57         58         59         60         13
SEHEN SIE HIER IHRE ZAHL ? JA
          16         17         18         19         20
          21         22         23         24         25
          26         27         28         29         30
          31         48         49         50         51
          52         53         54         55         56
          57         58         59         60         31
SEHEN SIE HIER IHRE ZAHL ? JA
          32         33         34         35         36
          37         38         39         40         41
          42         43         44         45         46
          47         48         49         50         51
          52         53         54         55         56
          57         58         59         60         46
SEHEN SIE HIER IHRE ZAHL ? NEIN
ICH WUERDE SAGEN, SIE HABEN SICH DIE ZAHL 27 GEMERKT.
NOCH EINMAL ? NEIN
```

Bild 2. Probelauf des Programms

einer mechanischen Ausgabeeinheit, etwa einer Teletype, läßt man die Instruktion am besten ganz weg.

Der für das Listing verwendete Drukker (eine IBM Selectric Mod. 735) enthält die Zeichen „<" und „>" nicht. Sie werden im System des Autors simuliert durch „((" bzw. „))" (siehe Zeilen 65 und 72).

Die mit einem Doppelpunkt versehenen Zeilennummern werden vom Compiler erzeugt und können ignoriert werden.

Obwohl BASIC hierfür eine denkbar ungeeignete Sprache ist, wurde versucht, die Methoden der strukturierten Programmierung anzuwenden. Der Hauptteil des Programms besteht aus wenigen Zeilen, die den Gesamtablauf beschreiben. Die eigentlichen Aktionen finden in den darauffolgenden Unterprogrammen (Subroutinen) statt. Es wurde streng darauf geachtet, daß jeder Programmteil genau einen Eingang und einen Ausgang besitzt. Solche Maßnahmen verleihen dem Programm eine klare, durchschaubare Struktur – eine wesentliche Voraussetzung, wenn man später Änderungen mit Erfolg und ohne unerwünschte Nebeneffekte anbringen will. Das Resultat dieser Bemühungen: das Programm funktionierte auf Anhieb fehlerfrei.

Mit Hilfe dieser Erläuterungen sollte es möglich sein, „1-aus-60" in praktisch jeden gebräuchlichen „BASIC-Dialekt" zu übertragen. Der Trick eignet sich sehr gut dazu, Ihren skeptischen Freunden zu demonstrieren, wie intelligent Ihr Computer ist.

AIM-65 sendet RTTY

Unter diesem Titel fanden Funkfernschreib-Amateure in Heft 16/1979 auf Seite 953 ein Programm, mit dem der AIM-65 direkt das für den Sender erforderliche Nf-FSK-Signal produziert. Dabei wurde von den genormten Frequenzen 1275 Hz (Space) und 2125 Hz (Mark) ausgegangen.

Zahlreiche Leser fragten nun bei uns an, was zu tun sei, wenn man statt 850 Hz Shift nur 170 Hz verwenden will, wie das auf Kurzwelle oft üblich ist. Der im *Bild* aufgelistete kurze Programmabschnitt gibt darüber Auskunft; die Space-Frequenz bleibt dabei 1275 Hz, und Mark wird 1445 Hz. Die Änderungen betreffen nicht nur die Frequenzparameter, die bereits in Heft 16/1979 genannt waren, sondern auch die Abfrage des Timers, ob für das nächste Bit ein Frequenzwechsel erforderlich ist.

```
<K>*=02B3
/08
 02B3 AD LDA A006
 02B6 90 BCC 02C3
 02B8 10 BPL 02CF
 02BA A9 LDA #58
 02BC 8D STA A004
 02BF A9 LDA #01
 02C1 D0 BNE 02CC
 02C3 F0 BEQ 02CF
```

Programmänderung für 170 Hz Shift des FSK-Signales. Da die Demodulation per Hardware geschieht, sind im Empfangsprogramm (s. Heft 15/1979) keine Änderungen erforderlich

Probleme traten offenbar auch oft beim Eingeben des Standardtextes auf. Dabei ist wie folgt vorzugehen:

1. AIM-Texteditor mit FROM=170 und TO=1F0 initialisieren. Nach IN = einfach Return drücken.
2. Text eintippen und mit zwei (!!!) Return-Tastendrücken beenden. Die maximale Textlänge beträgt etwa 100 Zeichen.
3. Escape drücken und das RTTY-Programm an der Adresse 0000 starten. Mit F1 läßt sich der eingegebene Standardtext beliebig oft aufrufen.

Leider ist es nicht möglich, das gesamte Programm auf einmal auf Kassette aufzuzeichnen. Vielmehr sollte man es in folgende Teile aufspalten: 0000...00AB; 0170...01F0; 0200...0300.

Manchmal wurde auch gefragt, wie es möglich ist, sendeseitig Mark und Space zu vertauschen. Hierzu ist lediglich an die Adresse 02B6 statt BCC hex B0 zu schreiben (BCS).

Ein Ausdruck der empfangenen Texte mit dem AIM-Printer ist prinzipiell leider nicht möglich. Hierfür bleibt dem Prozessor nämlich zwischen den einzelnen Zeichen nicht genug Zeit. Zu beachten ist schließlich noch, daß nicht alle ASCII-Zeichen ein Baudot-Äquivalent besitzen; unzulässige Zeichen führen zum Abbruch der Sendung, sobald sie aus dem Bufferspeicher ausgegeben werden sollen. Fe.

Ein Hex-Monitor in BASIC

Wenn ein BASIC-Programm zu langsam wird, weil ein Interpreter nicht gerade die schnellste Programmform darstellt, ist die Verwendung von Unterprogrammen in Maschinensprache sehr nützlich. Ein Programm, das die Eingabe hexadezimaler Maschinenbefehle und Daten, das Auflisten eines Maschinenprogramms und seinen Abruf gestattet, nennt man „Monitor".

Das in *Bild 1* aufgelistete BASIC-Programm ermöglicht es, auch in solche Computer Maschinensprache-Routinen „einzubauen", die über keinen solchen Monitor verfügen, z.B. die bisherigen Versionen des PET 2001, des TRS-80 und des ABC-80. Daß das Monitorprogramm selbst auf dem AIM-65 geschrieben wurde, der das wirklich nicht nötig hätte, braucht den Benutzer kaum zu kümmern, wenn man einmal von zwei Ziffern in der Zeile 600 absieht: Sie stellen die Sprungadresse für den USR-Befehl dar.

Der Umgang mit dem Monitorprogramm wird aus *Bild 2* deutlich. Das Auflisten eines Maschinenprogramms kann mit der Taste Q beschleunigt und mit S wieder verlangsamt werden; die Taste X gestattet das „Aussteigen" aus den Betriebsarten LIST und POKE. Das Beispiel in Bild 2 enthält ein Maschinenprogramm, das den Akku des 6502-Prozessors mit 33 (hex) und das Y-Register mit 56 lädt; dann erfolgt ein Rücksprung an die BASIC-Wiedereintritts-Adresse C0D1. Dieses Maschinenprogramm läuft zwar auf dem AIM-65, nicht aber auf allen anderen BASIC-Computern, weil diese andere Adressen oder gar eine andere CPU verwenden. Trotzdem wird die Arbeitsweise des Hex-Monitors recht deutlich; wie man erkennt, kann mit RUN direkt die Wirkung des Maschinenprogramms in Zusammenhang mit dem BASIC-Befehl USR überprüft werden.

Das Hex-Monitor-Programm ist zwar nicht sehr komfortabel, gestattet aber eine schnelle Eingabe auch längerer Maschinenprogramme und läßt sich sehr leicht an beliebige BASIC-Computer adaptieren. Fe.

```
10 REM ◆◆◆ HEX MONITOR ◆◆◆
20 REM COMMANDS: POKE,LIST,RUN
30 DIMH$(16):T=100
40 DATA0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
50 FORI=0TO15:READH$(I):NEXT
60 INPUT"CMD=";A$
70 INPUT"DEC.ADR.:FROM=";B
80 IFA$="LIST"GOTO200
90 IFA$="POKE"GOTO400
100 IFA$="RUN"GOTO600
110 GOTO60
200 INPUT"TO=";C
210 FORI=BTOC:D=PEEK(I)
220 E=INT(D/16):F=DAND15
230 PRINTI;H$(E)+H$(F)
240 FORJ=0TOT:GETD$
250 IFD$="Q"THENT=10
260 IFD$="S"THENT=200
270 IFD$="X"GOTO60
280 NEXTJ
290 NEXTI
300 GOTO60
400 PRINT" ":PRINTB;:GOSUB500
410 D=16◆I:GOSUB500
420 D=D+I:POKEB,D:B=B+1:GOTO400
500 GETD$:IFD$="X"THENCLEAR:GOTO30
510 IFD$<"0"ORD$>"F"GOTO500
520 FORI=0TO15:IFD$=H$(I)GOTO540
530 NEXT
540 PRINTD$;:RETURN
598 REM AIM USR ADR = 4, 5
599 REM PET USR ADR = 1, 2(CHANGE600)
600 POKE5,INT(B/256):POKE4,BAND255
610 INPUT"ARGUMENT";A
620 PRINT"RESULT:";USR(A):GOTO60
```

◄ Bild 1. Das hier abgedruckte BASIC-Listing entstand auf einem AIM-65, läuft aber, wenn nur die Zeile 600 etwas geändert wird, sofort auf anderen Systemen

```
CMD=? POKE
DEC.ADR.:FROM=? 3000
3000 A9
3001 33
3002 A0
3003 56
3004 4C
3005 D1
3006 C0
3007 CMD=? LIST
DEC.ADR.:FROM=? 3000
TO=? 3001
3000 A9
3001 33
CMD=? RUN
DEC.ADR.:FROM=? 3000
ARGUMENT? 0
RESULT: 13142
CMD=?
```

Bild 2. Ein Beispiel für den Umgang mit dem Hex-Monitor, wiederum mit dem AIM-65 programmiert (CPU: 6502)

„Richtige" Groß- und Kleinschreibung beim CBM-3032

Ein großer Vorteil des CBM-3032 gegenüber seinem kleineren Bruder PET-2001 ist die Tatsache, daß man ohne Shift-Taste kleine, mit Shift-Taste große Buchstaben schreibt – wie bei einer herkömmlichen Schreibmaschine. Die Ernüchterung kommt aber, wenn man versucht, gemischt groß und klein geschriebene Texte auszudrucken: Aus großen werden kleine Buchstaben und umgekehrt. Dieses Problem entsteht auch mit dem in Heft 1/1980 beschriebenen Basic-Texteditor.

Zur Richtigstellung von Klein- und Großschreibung beim Editor wurde deswegen eine spezielle Druckroutine geschrieben, die die Zeichen entsprechend umwandelt. Im *Bild* steht diese Routine in den Zeilen 970...995; sie gibt die Stringvariable C$ sowohl auf dem Bildschirm als auch auf dem Drucker aus. Die übrigen Zeilen wurden nur mit abgedruckt, um die erforderlichen Änderungen am Editor-Programm deutlich zu machen. Der Ausdruck erfolgt hier jetzt mit der Taste P, wobei man die Wahl hat, den Drucker mitlaufen zu lassen oder nicht. Fe.

```
READY.
660 PRINT"PRINTER? Y/N":R$=""
663 GETE$:IFE$=""GOTO663
665 IFE$="Y"THENOPEN1,7:PRINT#1,CHR$(2)
670 FORI=0TOD:IFA$(I)="END"THENPRINT:CLOSE1:RETURN
680 IFA$(I)>""THENGOSUB820
690 NEXTI
860 IFWDLEN(M$)THENC$=M$:GOTO970
920 C$=LEFT$(M$,K):GOSUB970
970 D$="":PRINTC$:IFE$<>"Y"THENRETURN
975 FORS=1TOLEN(C$):B$=MID$(C$,S,1)
980 IFASC(B$)>128THENB$=CHR$(ASC(B$)-128):GOTO990
985 IFASC(B$)>64THENB$=CHR$(ASC(B$)+128)
990 D$=D$+B$:NEXT
995 PRINT#1,D$:RETURN
```

Mit dieser Routine kann der in Heft 1 beschriebene Texteditor eine File ausdrucken, ohne Groß- und Kleinbuchstaben zu verwechseln

Nascom-Basic

Für den bekannten Mikrocomputer Nascom-1 wird jetzt ein 8-K-Basic-Interpreter in EPROMs und auf Kassette angeboten, der von der Software-Firma Microsoft stammt. Damit kommt er aus demselben Haus wie die Interpreter weitverbreiteter Systeme wie PET, TRS-80, Apple usw. Wie nicht anders zu erwarten, ist dieser neue Basic-Dialekt mit den vor ihm entstandenen Versionen weitgehend identisch. Dennoch hat er einige Eigenheiten, die erwähnenswert sind.

Der Interpreter arbeitet mit den Monitorprogrammen NASBUG T2, T4 und Nas-sys zusammen. Er belegt in der EPROM-Ausführung die Adressen E000...FFFF und in der Kassetten-Ausführung die Adressen 1000...2FFF.

NASCOM-Basic

Befehle	Funktionen	Kommandos
CLOAD*	ABS	CLEAR
CSAVE*	ASC	CLOAD
	AND	CLOAD?
CLS	ATN	CSAVE
DATA	COS	CONT
DEF	EXP	LIST
DIM	FRE	NEW
END	INT	NULL
FOR NEXT	LOG	RUN
GOTO	NOT	MONITOR
GOSUB	OR	WIDTH
IF THEN	POS	LINES
INPUT	RND	
LET	SGN	
ON GOTO	SIN	
ON GOSUB	SPC	
INP	SQR	
OUT	TAB	
PEEK	TAN	
POKE	USR	
DEEK	VAL	
DOKE	CHR$	
SET	LEFT$	
RESET	MID$	
POINT	RIGHT$	
PRINT	STR$	
READ	LEN	
REM		
RESTORE		
RETURN		
SCREEN		
STOP		
WAIT		

Drucker oder externe Datensichtgeräte können über eine Serienschnittstelle angeschlossen werden.

Formate

Neben Ganzzahlen sind auch 7stellige Gleitkommazahlen darstellbar, und zwar innerhalb eines Bereiches von etwa $10^{-38}...18^{+38}$. Stringvariable dürfen bis zu 255 Zeichen beinhalten. In Feldern kann man so viele Indizes verwenden, wie in eine Zeile passen, vorausgesetzt es ist der nötige Speicherplatz vorhanden. Der Index 0 ist zulässig.

Editierungsmöglichkeiten

Mit den Monitorprogrammen T2 und T4 sind die Editierungsmöglichkeiten relativ gering. Es können lediglich in einer Zeile die zuletzt eingegebenen Zeichen wieder gelöscht werden. Nachträgliches Einfügen oder Löschen von Zeichen ist nicht möglich. Die umfangreichen Editierbefehle des Nas-sys können aber voll genutzt werden, wenn man dieses Monitorprogramm verwendet. Damit braucht diese Basic-Version in bezug auf den Eingabekomfort bestimmt die Konkurrenz nicht zu fürchten.

Fehlermeldungen

Fehler werden sowohl im Direkt-Modus als auch in der indirekten Betriebsart (normaler Programmablauf) mit einem zweistelligen Buchstabencode angezeigt, der auf die Art des Fehlers hinweist. Wenn möglich, wird auch die entsprechende Zeilennummer angegeben. Nachdem ein Fehler aufgetreten ist, kehrt der Interpreter in die Kommandoebene zurück. Alle Variablenwerte bleiben aber erhalten, so daß man mit Hilfe einer GOTO-Anweisung das Programm fortsetzen kann.

Besondere Befehle und Kommandos

Alle Befehle, Kommandos und Funktionen sind in der Tabelle zusammengestellt. Die meisten sind von anderen Basic-Versionen her bekannt. Bei den Funktionen ist lediglich anzumerken, daß alle üblichen arithmetischen und logischen Funktionen sowie die meisten Stringoperationen möglich sind. An Befehlen und Kommandos tauchen allerdings einige neue auf. Übrigens ließen sich etliche davon in beide Spalten einreihen.

Neben Programmen lassen sich Daten mit CSAVE* und CLOAD* auf Kassette abspeichern und wieder einlesen. Den beiden Befehlen folgt jeweils der Name eines Feldes, das die gewünschten Daten enthält. Interessiert nur ein Teil eines mehrdimensionalen Feldes, dann besteht die Möglichkeit, nur bestimmte Zeilen einzulesen.

Mit WAIT kann man den Zustand eines Ports überwachen. Das Programm wartet so lange, bis ein bestimmter Zustand eingetreten ist. Den Befehlen PEEK und POKE entsprechen DEEK und DOKE. Der Unterschied besteht darin, daß bei den ersten beiden nur Zahlen bis 255 (entsprechend 8 Bit) als Argument auftreten dürfen, während bei DEEK und DOKE ein Bereich von –32 768...+32 767 (16 Bit) zugelassen ist. INP und OUT sind auch bei manchen anderen Systemen vorhanden. Mit ihnen liest man E/A-Ports bzw. gibt einen Wert aus.

In den Monitor kann man mit dem Befehl MONITOR springen. Von dort ist es jederzeit möglich, die Steuerung wieder an den Interpreter zu übergeben, auch ohne das gegenwärtige Basic-Programm zu zerstören.

Mit LINES gibt man an, wieviele Zeilen bei einem LIST-Kommando ausgegeben werden sollen. Normalerweise ist dieser Wert 5. WIDTH ist besonders nützlich, wenn man einen Drucker hat, der nicht die vorgegebenen 48 Zeichen/Zeile ausgeben soll. Mit diesem Befehl stellt man die gewünschte Zeichenzahl ein. Will man an einer bestimmten Bildschirmstelle schreiben, dann gibt man diese Position mit dem Befehl SCREEN an.

Anzumerken ist noch, daß für das neue Nascom-Basic ein sehr gutes Handbuch in deutsch existiert, das sich in erfreulicher Weise von den oft recht schlampigen Übersetzungen der amerikanischen Original-Literatur abhebt.

Rudolf Hofer

Björn Schwarz

Schachcomputer mit Niveau

Kaum ein Konsumartikel hat im zurückliegenden Jahr eine derart rasante technische Entwicklung durchgemacht wie der Schachcomputer. Die Schachprogramme wurden zum Teil erheblich verbessert und auch der Bedienungskomfort ist gestiegen. Von drei besonders interessanten Schachcomputern handelt der folgende Beitrag.

Computerspiele, allen voran die Schachcomputer, waren auch im vergangenen Jahr wieder stark gefragt. Waren die Geräte der ersten Generation teilweise noch mit schwerwiegenden Mängeln behaftet und ließ auch ihre Spielstärke oftmals zu wünschen übrig, so besitzen einige Neuentwicklungen bereits ein hohes Maß an künstlicher Intelligenz.

Seit dem ersten Vergleichstest in FUNKSCHAU 1979, Heft 8, S. 467–470 sind nur etwas weniger als 10 Monate vergangen, in denen sich jedoch auf dem Gebiet der Schachcomputer einiges getan hat. Konnte der Schachfan damals nur zwischen fünf verschiedenen Geräten von vier Herstellern auswählen, so bieten ihm nunmehr 5 Hersteller eine breite Palette von 14 verschiedenen Schachcomputern an. Selbstverständlich befolgen die meisten Geräte strikt die gültigen Schachregeln und beherrschen auch Spezialzüge wie „En Passant" und „Rochade", weisen aber dennoch bezüglich Spielstärke und Preis teilweise enorme Unterschiede auf.

Aus dem reichhaltigen Angebot stellen wir nachfolgend drei bemerkenswerte Schachcomputer vor, die jeweils eine gerätetypische, erstmals realisierte Besonderheit aufweisen (*Tabelle 1*).

Chess Challenger Voice

Der Urahne dieses Gerätes, der „Chess Challenger 3", war der erste auf dem deutschen Markt angebotene Schachcomputer (FUNKSCHAU 1978, Heft 21, S. 1045). Nachfolger dieses nach heutigen Maßstäben recht primitiv anmutenden elektronischen Schachspiels waren die bedeutend besser spielenden Geräte „Chess Challenger 10" und „Chess Challenger 7".

Die Krönung dieser in hohen Stückzahlen verkauften „Challenger"-Familie stellt der seit Ende Oktober 1979 erhältliche erste sprechende Schachcomputer „Chess Challenger Voice" (CCV) dar (*Bild 1*). Das Vokabular des mit 998 DM nicht gerade billigen Gerätes ermöglicht zwar keine tiefsinnige Konversation, enthält jedoch u. a. sämtliche Figuren und Spielfelder, die Bestätigung verschiedener Befehlseingaben, eine kurze Begrüßungsrede und die Ankündigung eines Schachmatts sowie das Eingeständnis einer Niederlage mit der lapidaren Feststellung „Ich habe verloren". Wohltönend tief ist diese synthetisch erzeugte Stimme schon, wenn sie auch sehr monoton klingt und Assoziationen zur Computerstimme der Fernsehserie „Raumschiff Enterprise" hervorruft.

Bei der ersten Konfrontation mit dem „Chess Challenger Voice" ist der menschliche Spielpartner zwar tief von dessen Sprechfähigkeit beeindruckt, wird sich dann aber fragen, ob ihm dieser Spaß tatsächlich 300 DM mehr wert ist, als der für 698 DM erhältliche „Chess Challenger 10", der ähnlich aufgebaut ist und auch eine vergleichbare Spielstärke aufweist. Auf den zweiten Blick wird der elektronikbegeisterte Schachspieler dann aber doch diese Zusatzausstattung zu schätzen wissen, wenn er nämlich bei Schachpartien mit höherer Spielstärke, wie

Tabelle 1: Hauptdaten der neuesten Schachcomputer

Eigenschaft	Chess Challenger Voice	Chess Champion Super System III	Sargon ARB/ MGS-System
spielt beide Farben	ja	ja	ja
spielt gegen sich selbst	ja	ja	ja
Anzeige von Alternativzügen	nein	ja	ja
Figurenanzeige in Symboldarstellung	nein	ja	ja
Stellungskontrolle während der Partie möglich	ja	ja	ja
Rücknahme von Einzelzügen	nein	ja (bis zu 10)	ja (bis zu 6)
Anzahl der Programmstufen	10	eingebaute Uhr	7
automatische Rochade	ja	ja	ja
automatisches „En Passant"	ja	ja	ja
automatische Bauernumwandlung	ja	ja	ja
Befolgung der 3-Zug- und 50-Zug-Remis-Regel	nein	ja	nein
Erkennung einer Patt-Situation	nein	ja	nein
Batteriebetrieb möglich	nein	ja	ja (x)
integriertes Schachspiel	ja	ja (xx)	ja
Preis (Stand 31.12.79)	998.– DM	398.– DM (xxx)	2998.–/ 998.– DM

Zeichenerklärung:
x nur MGS-System xx Zusatzgerät erforderlich xxx Grundgerät

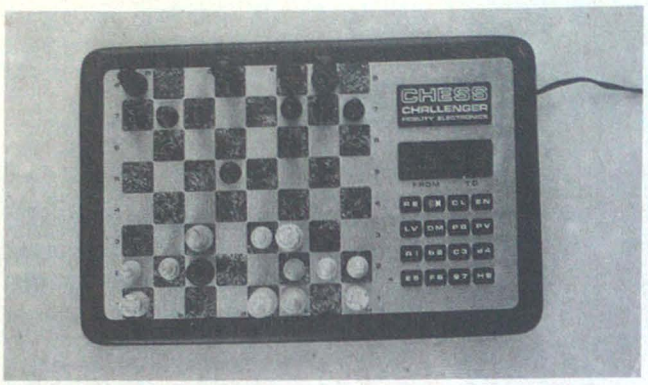


Bild 1. Chess Challenger Voice, der erste sprechende Schachcomputer

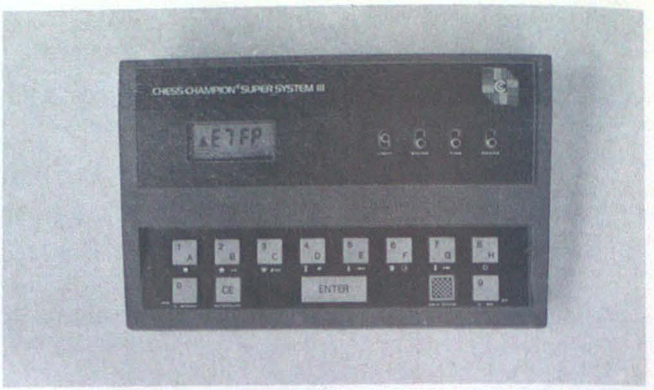


Bild 2. Chess Champion Super System III, der erste nach dem Baukasten-Prinzip aufgebaute Schachcomputer

z. B. Turnierschach und Problemschach, die Augen nicht mehr längere Zeit angestrengt auf die Anzeige richten muß, um den Antwortzug abzulesen. Statt dessen kann er eine andere Tätigkeit verrichten, bis er wieder durch den gesprochenen Gegenzug an das Schachbrett gebeten wird. Erste Versuche haben außerdem gezeigt, daß der „Chess Challenger Voice" in Verbindung mit einem Spezial-Schachspiel blinden Schachpartnern hervorragende Dienste leistet.

Neben der Sprechfähigkeit weist das Gerät u. a. folgende Merkmale auf: integriertes Schachbrett, 10 verschiedene Spielstärken mit „Denkzeiten" zwischen 1 s (Blitzschach) und mehreren Stunden (Problemschach), reichhaltige Eröffnungsbibliothek.

Chess Champion Super System III

Der seit Oktober 1979 angebotene Schachcomputer „Chess Champion Super System III" (*Bild 2*) stellt die wesentlich verbesserte Ausführung seiner Vorläufer „Chess Champion MK I" und „Chess Champion MK II" dar. Völlig neuartig an diesem Gerät ist das erstmals bei Schachcomputern angewandte Baukastenprinzip. Neben dem 398.– DM teuren Grundgerät für Netzbetrieb sind folgende Zusatzgeräte erhältlich: elektronisches LCD-Schachbrett (298.– DM), auf dem sämtliche Figuren mit natürlichen Symbolen dargestellt werden; elektronischer Speicher (179.– DM), der u. a. die Speicherung einer abgebrochenen Partie bis zu einem Jahr ermöglicht; Akkusatz (98.– DM) für den ca. 2–5stündigen netzunabhängigen Betrieb des Grundgerätes mit Speicher und LCD-Schachbrett; elektronischer Drucker (179.– DM), der jeden Spielzug auf einem Papierstreifen registriert und ein Attaché-Koffer (79.– DM) zur Aufbewahrung aller Geräte.

Das „Chess Champion Super System III" zeichnet sich durch eine gut ablesbare LCD-Anzeige aus, auf der nicht nur die Schachzüge, sondern auch die entsprechenden Figuren in Symboldarstellung und die international gebräuchlichen Zeichen für Schach, Schachmatt und das Schlagen einer Figur angezeigt werden. Das Gerät spielt sowohl mit den schwarzen als auch mit den weißen Figuren und schlägt auf Wunsch alternative Gegenzüge vor. Im Gegensatz zu seinen Vorgängern besitzt das „Chess Champion Super System III" eine eingebaute Uhr, so daß die Zeitdauer für jeden einzelnen Gegenzug des Schachcomputers exakt eingestellt werden kann. Weitere vorteilhafte Merkmale des „Chess Champion Super System III" sind u. a.: Erkennung einer Patt-Situation, Befolgung der 3-Zug- und 50-Zug-Remis-Regel, einfache Überprü-

Tabelle 2: Blitzschachpartie zwischen „Chess Champion Super System III" und „Chess Challenger Voice"

Zug-Nr.	Chess Champion Super System III (weiß)	Zeit (s)	Chess Challenger Voice (schwarz)	Zeit (s)
1	e2 – e4	1,2	c7 – c5	0,2
2	d2 – d4	1,3	d7 – d6	1,4
3	Lf1 – b5 +	1,1	Lc8 – d7 1111	5,6
4	Lb5 x d7 +	1,3	Sb8 x d7 1111	2,7
5	Sb1 – c3	1,2	Sg8 – f6	1,4
6	Lc1 – f4	1,4	e7 – e5	1,4
7	d4 x e5	1,3	d6 x e5	1,4
8	Lf4 – e3	1,4	Dd8 – a5	1,4
9	Dd1 – d3	1,8	Da5 – b4	1,6
10	Ke1 – c1 o-o-o	1,4	Lf8 – e7	1,7
11	Sg1 – f3	1,3	c5 – c4	1,9
12	Dd3 – e2	2,0	e8 – g8 o-o	1,9
13	h2 – h4	1,3	Le7 – c5	2,2
14	Le3 x c5	1,4	Db4 x c5	2,1
15	De2 – e3	1,4	Ta8 – c8	2,1
16	De3 x c5	1,3	Tc8 x c5	1,4
17	Td1 – d6	1,3	Tf8 – e8	1,2
18	Th1 – d1	1,2	Sd7 – b6	1,3
19	Kc1 – b1	1,3	Tc5 – a5	1,4
20	g2 – g3	1,3	Sf6 – g4	1,4
21	Td6 – d8	1,3	Te8 x d8	1,2
22	Td1 x d8 #	0,9	„ich habe verloren"	0,3
	Gesamtzeit	29,4 s		37,2 s

Zeichenerklärung:

–	ziehe Figur von... nach...		
x	schlage Figur	1111, 2222, 3333	Chess Challenger Voice analysiert eine neue strategische Variante
+	Schach		
#	Schachmatt	1111	zweite strategische Variante
o-o	kurze Rochade	2222	dritte strategische Variante
o-o-o	lange Rochade	3333	vierte strategische Variante

fung der Figurenpositionen, einfache Aufstellung von Spielsituationen (Problemschach) und die leicht durchführbare Rücknahme von bis zu 10 Einzelzügen (nur in Verbindung mit dem Speichermodul möglich).

„Chess Champion Super System III" weist gegenüber den meisten anderen derzeit erhältlichen Schachcomputern eine weitere Besonderheit auf: das verwendete Computerschachprogramm, das von einem der besten Computerschachspezialisten der Welt entwickelt wurde, arbeitet nicht mehr ausschließlich nach der „Brute-Force"-Methode, sondern bedient sich zusätzlich des „Alpha-Beta-Algorithmus". Die dadurch erzielte Steigerung der Spielstärke ist beachtlich. Ein nach der Brute-Force-Methode arbeitender Schachcomputer untersucht *alle* möglichen Varianten bis zu einer bestimmten Tiefe und ermittelt dabei den Zug mit der größten Wertigkeit. Beim „Alpha-Beta-Algorithmus" handelt es sich um einen Zusatz zur „Brute-Force"-Strategie, durch den die Suche nach dem besten Zug erheblich beschleunigt wird. Dies wird dadurch erreicht, daß eine Vielzahl von Zügen und daraus resultierende Zugfolgen nicht mehr berücksichtigt werden, wenn sie keine ausreichende Bewertung aufweisen.

Bei Testspielen erwies sich das „Chess Champion Super System III" als starker Gegner, der leider über kein variantenreiches Eröffnungsspiel verfügt, jedoch in den entscheidenden Phasen des Mittel- und Endspiels überdurchschnittlich gut reagierte. Die vorgelegten Problemschachaufgaben wurden ebenfalls innerhalb kurzer Zeit gelöst.

Bild 3. Der Autor beim Spiel gegen „Sargon ARB", dem ersten „sehenden" Schachcomputer

Sargon ARB (Automatic Response Board)

Handelt es sich beim „Chess Challenger Voice" um den ersten sprechenden Schachcomputer, so gebührt dem „Sargon ARB" das Prädikat, der erste „sehende" Schachcomputer zu sein. Aufgrund dieser überragenden Eigenschaft ist es nicht mehr erforderlich, die Spielzüge über eine Tastatur einzugeben; die entsprechenden Befehle werden statt dessen durch kleine Reed-Kontakte übermittelt, die sich in jedem der 64 Felder des Spezialschachbretts befinden. Die Betätigung der Reedkontakte erfolgt durch das Magnetfeld kleiner, in den Sockel der Figuren integrierter Dauermagnete.

Obwohl Mitte Dezember 1979 noch kein serienmäßiges Gerät, sondern lediglich ein Prototyp existierte (*Bild 3*), konnten mit diesem bereits reichhaltige Erfahrungen gesammelt werden. „Sargon ARB" erwies sich bei einer Vielzahl von Testspielen als ernstzunehmender Gegner, bedingt durch das ebenfalls nach dem „Alpha-Beta-Algorithmus" operierende Computerschachprogramm Sargon 2,5. Bei diesem Programm handelt es sich um eine Verbesserung des vom kalifornischen Computerschachexperten Dr. Dan Spracklen und seiner Ehefrau Kathe entwickelten Schachprogramms Sargon 2, das bei den im Dezember 1978 in Washington ausgetragenen nordamerikanischen Computerschach-Wettbewerben einen hervorragenden 3. Platz erreichte und dabei u. a. sogar den 5 Millionen Dollar teuren professionellen Laborcomputer Awit besiegte.

Zum Erwerb des „Sargon ARB" müssen Liebhaber zwar tief in die Tasche greifen (2998.– DM), erhalten dafür aber auch ein Gerät, das bezüglich Spielstärke und Bedienungskomfort keine Konkurrenz zu scheuen braucht. Ein billigeres Gerät, das ebenfalls das Schachprogramm Sargon 2,5 verwendet, ist das seit den letzten Dezembertagen des Jahres 1979 zum Preis von 998.– DM erhältliche MGS (Modular Game System)-Multispielsystem mit auswechselbarer Spielkassette. Sowohl der „Sargon ARB" als auch das MGS-System geben ihren Gegenzug nicht mehr wie ihre Vorgänger nach Ablauf der eingestellten Antwortzeit bekannt, sondern verfügen über sieben verschiedene wählbare Spielstärken; das MGS-System nutzt darüber hinaus die Denkzeit des Gegners für die Ermittlung seines Gegenzuges mit aus. Weitere Vorzüge der beiden Geräte sind u. a.: Leichte und schnelle Aufstellung von Spielsituationen, einfache Überprüfung der Figurenpositionen, Unterbreitung eines Zugvorschlages an den Gegner, Rücknahme von bis zu 6 Einzelzügen.

Tabelle 3: Turnierschachpartie zwischen „Chess Challenger Voice" und „Sargon ARB"

Zug-Nr.	Chess Challenger Voice (weiß)	Zeit (s)	SargonARB (schwarz)	Zeit (s)
1	d2 – d4	0,5	Sg8 – f6	0,5
2	c2 – c4	0,5	g7 – g6	0,5
3	Sb1 – c3	0,5	Lf8 – g7	0,5
4	e2 – e4	0,5	Sb8 – c6	76,0
5	e4 – e5 1111	11,5	Sf6 – g8	78,0
6	Sg1 – f3 1111	15,0	d7 – d6	76,0
7	Dd1 – e2 1111	9,5	Lc8 – g4	35,0
8	e5 x d6 1111, 2222	66,0	Sc6 x d4	43,0
9	d6 x c7 1111, 2222	85,0	Dd8 – d7	1,0
10	De2 – e4 1111, 2222	79,0	Lg4 – f5	1,0
11	c7 – c8D 1111, 2222	98,0	Ta8 x c8	1,0
12	Sf3 – e5 1111, 2222	130,0	Lg7 x e5	1,0
13	De4 x e5 1111, 2222	65,0	Sd4 – c2 +	35,0
14	Ke1 – e2	1,5	Dd7 – d3 #	1,0
15	„ich habe verloren"	1,0		
Gesamtzeit		9 min 23,5 s		5 min 49,5 s

Vergleichstest

Zur Ermittlung der Spielstärken der einzelnen Geräte wurden mehrere Schachpartien zwischen ihnen ausgetragen. Die ersten Spiele bestritten der „Chess Challenger Voice" und das „Chess Champion Super System III" gegeneinander. Von 4 Blitzschachpartien gewann der „Chess Challenger Voice" nur eine einzige, zwei gewann das „Chess Champion Super System III" (*Tabelle 2*) und trotzte seinem Gegner in der vierten Partie trotz dessen großer Überlegenheit noch ein Patt ab. Bei den beiden nachfolgenden Turnierspielen, in denen entsprechend dem Reglement die ersten 40 Züge jedes Kontrahenten innerhalb von 2 Stunden erfolgen müssen, waren beide Geräte jeweils einmal erfolgreich.

In der nächsten Spielrunde traten der „Chess Challenger Voice" und der „Sargon ARB" gegeneinander an. Das Ergebnis war eindeutig: „Sargon ARB" gewann sämtliche 6 Blitzschachpartien und auch die beiden Turnierspiele (*Tabelle 3*). Der „Chess Challenger Voice" schien dabei von der Leistungsfähigkeit seines Gegners derart beeindruckt zu sein, daß es ihm zeitweilig die Sprache verschlug. Als Ursache für diesen Defekt erwies sich das für den Dauerbetrieb etwas zu schwach dimensionierte Netzteil.

Die überlegene Spielstärke des „Sargon ARB" ließ auch in der letzten Spielrunde seinem Gegner „Chess Champion Super System III" keine Chance. Sowohl die Blitzschachpartie als auch die nachfolgenden vier Turnierschachpartien konnte der „Sargon ARB" für sich entscheiden (*Tabelle 4*).

Nach diesem eindeutigen Ergebnis können sowohl „Sargon ARB" als auch das nach demselben Schachprogramm operierende „MGS-System" als die derzeit spielstärksten Heim-Schachcomputer der Welt bezeichnet werden. In unserer schnellebigen Zeit dürfte es jedoch nur eine Frage der Zeit sein, bis auch das Schachprogramm „Sargon 2,5" seinen Bezwinger findet.

Tabelle 4: Turnierschachpartie zwischen „Chess Champion Super System III" und „Sargon ARB"

Zug-Nr.	Chess Champion Super System III (weiß)	Zeit (s)	Sargon ARB (schwarz)	Zeit (s)
1	d2 – d4	1,5	d7 – d5	1,0
2	Sb1 – c3	1,5	Sb8 – c6	74,0
3	Lc1 – f4	74,0	e7 – e6	1,0
4	Dd1 – d3	1,5	Sc6 – b4	36,0
5	Dd3 – b5 +	36,0	Lc8 – d7	41,0
6	Db5 x b7	41,0	Sb4 x c2 +	1,0
7	Ke1 – d2	1,5	Sc2 x a1	1,0
8	Db7 x c7	1,5	Dd8 x c7	67,0
9	Lf4 x c7	1,5	Sg8 – f6	1,0
10	Sg1 – f3	1,5	Lf8 – b4	41,0
11	(e2-e3) Kd2 – d3	41,0	e8 – g8 o-o	25,0
12	Lc7 – e5	25,0	Lb4 x c3	21,0
13	b2 x c3	1,5	Ld7 – b5 +	1,0
14	Kd3 – d2	1,5	Sf6 – e4 +	37,0
15	(Kd2-d1) Kd2 – e1	37,0	Sa1 – c2 +	39,0
16	Ke1 – d1	1,0	Se4 x f2 +	2,0
17	Kd1 x c2	1,5	Lb5 – a4 +	47,0
18	Kc2 – b2	1,5	Sf2 x h1	41,0
19	e2 – e3	1,5	Sh1 – f2	1,0
20	Lf1 – e2	1,5	Tf8 – e8	1,0
21	Kb2 – a3	1,5	La4 – d1	56,0
22	Le2 – b5	1,5	Te8 – c8	57,0
23	Ka3 – b4	1,5	a7 – a5 +	34,0
24	Kb4 – a3	1,0	Ld1 x f3	1,0
25	g2 x f3	1,5	Tc8 x x3 +	25,0
26	Ka3 – a4	1,5	Sf2 – d1	1,0
27	a2 – a3	1,5	Sd1 – b2 #	1,0
28	LOSE	1,5		
Gesamtzeit		4 min 46 s		10 min 54 s

Zeichenerklärung:
( ) in Erwägung gezogener, jedoch nicht ausgeführter Zug 22499 15937

Nachtrag

Computer-Clubs

FUNKSCHAU 1979, Heft 24, Seite 1422

In unserer Aufstellung fehlte ein Club, der offenbar auch in Deutschland zahlreiche Mitglieder hat: der Personal Programmers' Club. Herausgeber ist Richard Nelson, 2541 W. Camdon Place, Santa Ana, CA 92704, USA. Der Bezugspreis für das damit verbundene PPC-Journal beträgt in Europa 23 US-Dollar jährlich; die Mitgliedschaft kann im Januar, Mai oder September beginnen. Offenbar werden aber Bestellungen nur dann angenommen, wenn ihnen der Mitgliedsbeitrag z. B. in Dollarnoten beigelegt wurde.

Fälschlicherweise wurde beim TI-58/59-Software-Club ein Mitgliedsbeitrag von 42 DM genannt; richtig muß es 20 DM heißen.

Auch in Mainz hat sich ein Computer-Club etabliert; derzeit hat er etwa 30 Mitglieder. Vorsitzender ist Klaus Straka, Westring 46, 6500 Mainz.

Der mittlerweile über 2000 Mitglieder zählende Schweizer Computerclub gibt eine Zeitschrift namens „Hobby- und Kleincomputer" heraus. Interessenten können sich an Ernst Erb, Seeburgstr. 18, CH-6002 Luzern, wenden.

Österreichische Mikrocomputer-Interessenten können sich an den Computer-Hobby-Club, Eisvogelstraße 4, 1060 Wien, wenden. Die Mitgliedschaft kostet 400 Schilling im Jahr, für Studenten und Schüler nur 200 Schilling. Der Club existiert seit Dezember 1979.

Basic-Texteditor

FUNKSCHAU 1980, Heft 1, Seite 87

In das Programm (Bild 1) wurde statt der Kassetten-Leserroutine versehentlich die Floppy-Routine eingebaut. Für das Lesen von Files von der internen PET-Kassette müssen folgende Zeilen geändert werden:

540 INPUT „FILENAME="; M$: OPEN 1, 1, 0, M$

550 Der Zeileninhalt nach dem Doppelpunkt kann entfallen

570 CLOSE1: PRINT „LOADED": RETURN

Die Routine in Bild 1 ab Zeilennummer 700 ist bereits für den Kassettenbetrieb ausgelegt und braucht nicht geändert zu werden.

Das Editorprogramm benutzt mehrmals die Befehlsform RIGHT$ (A$(J), LEN (A$ (J)) – 1). Sie läßt sich ein wenig vereinfachen, wenn man den MID-Befehl verwendet: MID$ (A$(J), 2).

Das Capricorn-Projekt

Bereits im letzten Heft war nachzulesen, daß Hewlett-Packard's lang gehütetes Capricorn-Projekt – so wurde der HP-85 firmenintern genannt – in Kürze lieferbar ist. Hier ein erster Erfahrungsbericht.

Vor etwas mehr als einem Jahr wurde durch vorsichtige Andeutungen in amerikanischen Fachzeitschriften der Öffentlichkeit erstmals bekannt, daß sich der renommierte Meßgeräte- und Tischcomputer-Hersteller Hewlett-Packard in Bälde auch auf dem Hobby- und Kleinfirmenmarkt zu tummeln gedenke. Da es zum Stil des Hauses HP gehört, Produkte erst dann anzukündigen, wenn die Serienfertigung bereits begann, hüllten sich Firmensprecher bis zuletzt in Schweigen, wenn sie nach den Eigenschaften der kommenden Maschine gefragt wurden. Kein Wunder, daß sich um HPs ersten „Personal Computer" bereits vor seiner Vorstellung die buntesten Gerüchte rankten; was aber amtlich bekannt wurde, das war nicht viel mehr als die durch eine Indiskretion vorzeitig ans Licht gekommene Tarnbezeichnung des Rechners „Capricorn", das bei HP hausintern verwendete Schlüsselwort für das neue Produkt. Wer mit „Capricorn" zu tun hatte, mußte sich verpflichten, bis zur Vorstellung des Rechners gegenüber jedermann zu schweigen.

War diese Geheimniskrämerei angemessen? Kommt mit dem HP-85 ein Gerät auf den Markt, das sich wesentlich von Konkurrenzprodukten unterscheidet, oder sollte hier nur eine abgemagerte Version der hinlänglich bekannten Tischrechner aus dem gleichen Hause angeboten werden? Nichts dergleichen. Das Fazit aus vielen Abenden und Wochenenden Spielens, Übens und später Arbeitens mit dem neuen Computer berechtigt zu der Formulierung, daß sich mit dem HP-85 eine neue Dimension für die Freunde des Computer-Steckenpferds auftut, daß eine eigene Geräteklasse erschlossen wird, die es bisher im Bereich des „Personal Computing" noch nicht gab: Die Klasse der kompromißlos in kommerzieller Qualität gefertigten Rechner.

Qualität hat ihren Preis

Qualität hat ihren Preis – das gilt auch dann, wenn ein Erzeugnis in jenen Markt zielt, in dem der Anschaffungspreis weit eher ein Kaufhindernis ist als bei Kunden, die ihren Rechner kommerziell nutzen. Vom Deutschland-Preis der HP-85 war beim Entstehen dieses Beitrages nur soviel bekannt, daß er voraussichtlich irgendwo zwischen

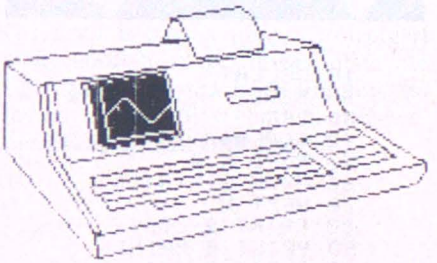


Bild 1. Ein Star porträtiert sich selbst: Hier hat der neue Stern am Computer-Himmel seinem Thermodrucker den Auftrag gegeben, ein Selbstbildnis anzufertigen

5000 und 6000 Mark liegen werde: In einer Größenordnung mithin, die den Rechner allenfalls für Schulen, Clubs und betuchtere Privatleute erreichbar erscheinen läßt. Freilich sollte man sich bei Preisvergleichen nicht täuschen lassen: Wenn man die beim HP-85 eingebauten Zusatzeinrichtungen wie Schirmausgabe, plottender Thermodrucker mit Spaltenzahl 32 und schnelles Datenkassettenlaufwerk mit komfortablem Betriebssystem den Preisen von PET, TRS-80, ITT-2020 und Apple hinzurechnet, wenn man weiterhin ins Auge faßt, daß einige HP-Produkte im Laufe der Zeit fast auf den halben Preis zurückgingen, dann verliert die oben genannte Zahl einiges von ihrer Schockwirkung.

Nach dieser summarischen Wertung zum Konkreten: Der Rechner kam in einer schreibtischhohen Riesenkiste, in der das nur schreibmaschinengroße Gerät trickreich in Kunststoffrahmen aufgehängt war – zusammen mit einigen hundert Seiten vorzüglicher Dokumentationsmaterials. Die Inbetriebnahme scheiterte erst einmal daran, daß der laut Handbuch einzufügende Sicherung-Einsatz vorläufig unauffindbar schien, bis er schließlich im Papierfach des Druckers entdeckt wurde. Es lagen zwei Datenkassetten bei, eine unbespielt, die andere mit sechszehn Spiel- und Nutzprogrammen versehen, darunter eines, das den Rechner minutenlang trillerreiche Barockmusik brummeln und pfeifen ließ. Ein fröhlicher Einstand!

Der Autor hatte, was die Programmiersprache angeht, dem Rechner mit einigem Unbehagen entgegengesehen, da er als Freund von Microsoft-Basic der älteren, komplizierten und bisweilen vergleichsweise umständlichen Sprachversion HP-Basic in der Vergan-

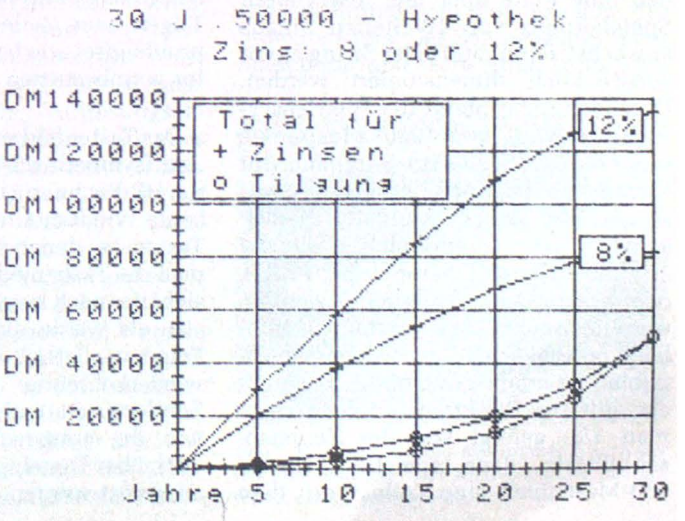
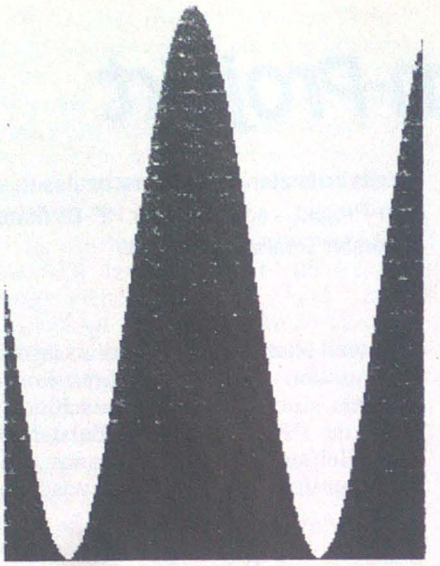


Bild 2. Häusle-Bauern steht der Rechner auch grafisch zur Seite: Hier malt er aus, was einem frischgebackenen Eigenheimbesitzer in den nächsten 30 Jahren finanziell so blüht

genheit nicht viel Liebe abgewinnen konnte – und hier gab es schon eine erste Überraschung. Zwar ist die Sprache des HP-85 schon aus Gründen der Kompatibilität, wie erwartet, eng an Eigenheiten des Hauses Hewlett-Packard orientiert, jedoch wurden Neuerungen aufgenommen, die dem Anhänger der „Hobbycomputer-Weltsprache Nummer Eins" *Microsoft-Basic* den Zugang erleichtern: Ein ausgeklügeltes System von „Default-Values" beim Programm-Schreiben läßt dem Benutzer die Wahl, ob er trickreich oder schlicht zu programmieren gedenkt – solange er nur einige Besonderheiten beachtet.

Komfort mit Wermutstropfen

Mehrfachstatements müssen, höchst ungewohnt, durch das „At-Zeichen" getrennt werden, im Gefolge von „IF...THEN..." – Statements mag der Interpreter gewisse Anweisungen nicht sehen, Variable müssen wesentlich lakonischer benannt werden als in den Microsoft-Dialekten von 2020, PET, TRS-80 und Genossen, und schließlich – höchst bedauerlich und aus des Autors Sicht ein empfindlicher Mangel – kann der HP-85 nicht mit Datenfeldern umgehen, deren Elemente Zeichenketten sind: Bei numerischen Arrays verkraftet der Rechner im Höchstfall zwei Dimensionen. Eine Belastung aus der Verbindung mit seinen älteren Brüdern stellt für den Computer auch die Tatsache dar, daß Zeichenketten nicht dynamisch verwaltet werden: Ein String belegt beim HP-85 im Speicher den Raum einer Zeichenkette mit 18 Elementen – gleichgültig, ob die Kette allein aus dem Zeichen „." oder einem Text wie „AUF, HOLTERDIPOLTER" besteht. Mag der Programmierer diese Speicherraumvergeudung nicht in Kauf nehmen oder ist damit zu rechnen, daß eine Kette über den reservierten Speicherplatz von 18 Stellen hinaus anwächst, dann muß jeder String zuvor ausdrücklich dimensioniert werden: Durchaus nicht Stand der (Betriebssystem-) Technik von 1980! Pluspunkte sammelt der HP-85 im Vergleich mit Microsoft-Maschinen hingegen, was seine Betriebsart „Calculator Mode" angeht. Hier wollen Rechner wie der 2020 oder der PET nämlich per PRINT oder Fragezeichen vorbereitet werden, wenn man nach dem Programmdurchlauf von Hand den Inhalt einzelner Variablen zu erfahren wünscht. Das gleiche gilt für Direktrechnungen. Tippt man 1/2, gefolgt von der Zeilenabschlußtaste, dann speichern Microsoft-Maschinen eine Zeile 1 mit dem



```
10 GCLEAR
20 SCALE -1,1,0,600
30 DEG
40 FOR N=0 TO 600
50 MOVE N,0
60 DRAW SIN(N),N
70 NEXT N
80 PRINT @ COPY
90 PRINT @ PRINT
100 PLIST @ END
```

Bild 3. Wer nur Standard-Basic oder einen der Microsoft-Basicdialekte kennt, dem wird Ungewohntes an der Programmiersprache des HP-85 auffallen: Hier ein Funktionsausdruck und die dazugehörige Programmschleife

(unsinnigen) Befehl /2. Der '85 dagegen erkennt in solchen Fällen selbsttätig, daß hier unmittelbar gerechnet werden soll, und antwortet brav mit .5 – sehr schön!

Makellos ist auch der Editor. Das Redigieren von Programmen weist keine der Umständlichkeiten von 2020, Apple oder TRS-80 auf, Einfügen und Tilgen von Zeilenteilen erfolgen so problemfrei wie beim PET, dessen Editor ja unbestritten Maßstäbe setzte.

Das Tastenfeld selbst ist so solide und „griffsympathisch", wie bei den teureren HP-Rechnern; angenehm die eingebaute Wiederholungsfunktion all jener Tasten, bei denen dies sinnvoll ist: Läßt man die Taste niedergedrückt, so wirkt sich dies nach kurzer Umschaltpause so aus, als werde die Taste in schneller Folge wiederholt betätigt. Für häufig wiederkehrende Anweisungen sind Sondertasten vorhanden, jedoch können die entsprechenden Funktionen auch über Texteingabe über den Schirm ausgelöst werden.

Platten auf 49 000 Punkten

Der Bildschirm – Diagonale rund 13 Zentimeter – bietet Raum für 16 Zeilen zu je 32 Zeichen. In der Betriebsart „Grafik" können über 49000 Bildpunkte einzeln gesetzt oder gelöscht werden (192 Zeilen zu je 256 Spalten); dies entspricht dem Auflösungsvermögen von ITT 2020 und Apple, mit dem Unterschied freilich, daß beim '85 Farbgrafiken nicht möglich sind und der Bildaufbau deutlich langsamer erfolgt. Animation durch pseudokinematografische Effekte, wichtig für Bewegungsspiele, beherrscht der neue Computer deswegen nur bedingt. Dafür stellt das Betriebssystem der HP-Maschine beim Plotten und Zeichnen jedoch ungleich mehr Hilfsroutinen zur Verfügung: Die Anfertigung von Graphen exotischer Funktionen hat dem Autor immer wieder Stunden voller Vergnügens geschenkt – dies um so mehr, als auch der Drucker über das gleiche Auflösungsvermögen verfügt wie der Schirm.

Bemerkenswert ist auch vor allem die Datenkassettenstation, die sich als ein Mittelding zwischen Floppystation und Nf-Kassettenwerk erweist: Kapazität über 200 Kilobyte bei einer Rückspulzeit von weniger als einer halben Minute, Zugriff auf Programme oder Daten selbsttätig über den Namen der Aufzeichnung. Mit Hilfe dieses einzigen Laufwerks, den Befehlen „CHAIN" (Verketten von Programmen) und der Variablendefinition „COM" (Bewahren von Werten und Zeichen für Folgeprogramme) kommt man zu Bearbeitungsmöglichkeiten von Jobs unbegrenzten Umfangs. Freilich, auch hier ein Pferdefüßchen: Datenkassetten sind viel, viel teurer als das bedeutend schnellere Massenspeichermedium Floppy-Disk oder Compactcassetten.

Haushoch überlegen gegenüber allen anderen Personal-Computern ist der HP-85, was Rechengenauigkeit (12 signifikante Stellen) und Rechenbereich angeht. Der Bereich des Variablentyps REAL reicht von plus/minus 9,99999999999 mal zehn hoch plus/minus 499! Ungenauigkeiten und Rundungsfehler, wie sie vom Verfasser bei ITT-2020, PET, Apple und PC 100 beobachtet wurden, konnten beim '85 nicht provoziert werden: Ein Vorteil der BCD-Repräsentation von Realzahlen bei Hewlett-Packard; die anderen Rechner bilden ihre „REALS" rein binär ab.

Hilfreich zur Hand geht der Rechner seinem Benutzer nicht nur beim Programmieren und im Betrieb, sondern auch bei der Fehlersuche und bei Pro-

grammreparaturen. Während der Bearbeitung eines Kundenprogramms kann der Rechner Wertänderungen bei Variablen melden, Sprünge anzeigen und Analyseprotokolle drucken, die das Fehlersuchen zum Vergnügen machen. Ein leistungsfähiges Renumerierungsverfahren und selbsttätiges Setzen von Statement-Nummern erleichterte das Programmieren deutlich: Auch hier – Spitzen-Komfort!

Von vielen weiteren Eigenschaften des HP-85, die ihn zum gegenwärtigen Spitzengerät der Computer unter 7000 Mark machen, sei – stellvertretend für alle anderen – nur noch eine genannt, die allerdings verblüfft: Der Rechner kann, quasi gleichzeitig, mehrere Programme bearbeiten. Mit Hilfe eingebauter TIMER-Funktionen läßt sich der Rechner nämlich zwischen verschiedenen Jobs hin- und herschalten. Gleichzeitig musizieren, zeichnen und rechnen: Das macht dem HP-85 keiner seiner Konkurrenten so leicht nach!

Kein PEEK und POKE

Kommen wir zum Schluß-Urteil nach knapp 80 Stunden Betriebserfahrung: Der HP-85 zeigt Eigenschaften, die ihn zur Traum- und Wunschmaschine für Hobbyprogrammierer machen und die ihm auch bei kommerziellen Benutzern, im Ausbildungswesen und vor allem im Schulbetrieb gute Chancen schaffen, zum Maßstab zu werden, an dem sich andere Produkte künftig werden messen lassen müssen. Wer freilich gern mit anderen Sprachen als Basic arbeitet, wer sich die Wahl offenhalten will, ob er ein Problem per Assembler, FOCAL, XPLO oder PASCAL angeht: Der ist mit längst nicht so soliden und bei weitem nicht so komfortablen, dafür aber softwaremäßig bedeutend flexibleren Maschinen wie Apple-II oder ITT-2020 besser bedient. HP weigert sich nämlich strikt, etwas über die CPU-Assemblersprache verlauten zu lassen, und der HP-85 enthält auch weder PEEK- noch POKE-Befehle.

Drucker-Verwirrung

Beim Abdruck unseres AIM-65-Programms in Heft 25/1979 zur Simulation der CPU 8080 entstand bei manchen unserer Leser, zahlreichen Anrufen nach zu schließen, der Eindruck, als gäbe es zwei unterschiedliche AIM-65-Drucker-Zeichencodes: Normalerweise liefert der AIM-65 ja durchgestrichene Nullen. Nein! Sie können ganz beruhigt sein: Alle AIM-65-Ausdrucke fertigen wir normalerweise mit einem Silent-700-Terminal von Texas Instruments an, dessen Thermodrucker Nullen ohne Querstrich produziert. Das Silent-700 läßt sich problemlos über die 20-mA-Schnittstelle an den AIM-65 anschließen.

2650-Tutor

Der Mikrocomputer „Instructor 50"

Der Instructor 50 gehört nicht zu den preiswertesten Einplatinencomputern. Er hat aber einige Vorzüge, die das Arbeiten mit ihm sehr erleichtern, und wird im Gehäuse einschaltfertig geliefert.

Man braucht das mitgelieferte Steckernetzteil nur noch in die Steckdose zu stecken, schon meldet sich der Computer mit „HELLO" auf seinen Siebensegmentanzeigen. Sogar für den Kassettenrecorderanschluß ist gesorgt, ein Verbindungskabel wird mitgeliefert. Nun gleich die mitgelieferte Kassette in einen Recorder eingelegt und schon kann's losgehen. Wenn Sie täglich AFN hören, kein Problem – sollten Sie jedoch diese Gelegenheit bisher noch nicht genutzt haben, werden Sie mit dem „Slang" des Sprechers Ihre (Anfangs-)Schwierigkeiten haben. Nach einer kurzen Erläuterung über das Wunderwerk Mikroprozessor folgt eine Einstellanweisung für den Kassettenrecorder und dann auch schon das erste Programm: „Billboard" – eine Laufschrift.

Anhand der wirklich umfangreichen mitgelieferten Literatur kann man die einzelnen Programmschritte (mit Erläuterungen) verfolgen. Man erhält 3 Bücher: Software Applications Manual (98 Seiten), „Introduction to the Instructor 50 desktop Computer" (ca. 65 Seiten) und den Users' Guide (ca. 300 Seiten).

Die Eingabetastatur ist in 2 Teile gegliedert: 12 Funktionstasten, daneben 16 Hex-Tasten. Die Tasten haben zwar keinen „Druckpunkt", sind aber trotzdem recht gut zu handhaben. Die weiß aufgebrachte Schrift zeigt jedoch nach längerem Gebrauch Auflösungserscheinungen – ein Handicap bei vielen µP-Systemen. Als nützlich können sich auch die 8 Schalter erweisen, mit denen man die parallelen I/O-Ports „von Hand" setzen kann. Der Zustand der Ports kann mit LEDs kontrolliert werden. Mit einem weiteren Schalter können 3 I/O-Adressen angewählt werden (Memory 0FF, Extended I/O Port 07 und Non-Ext. Data Port). An der Unterseite(!) befindet sich ein weiterer Schalter: Interrupt Selector. Wird dieser Schalter von der Stellung „Keybord" auf „AC-Line" gestellt, so wird alle 20 ms ein Interrupt erzeugt, eine Hilfe bei Echtzeit-Steuerungsaufgaben.

Der Instructor 50 bietet noch ein nützliches Bonbon: Über den Tasten befindet sich eine Tabelle, in der alle Befehle im Hex-Code und den mnemonischen Abkürzungen aufgeführt sind; für den fortgeschrittenen Programmierer eine wertvolle Hilfe.

Nun zur Technik: Als CPU wird der 2650 eingesetzt, ein universell einsetzbarer, statischer Prozessor mit 8 bit Wortlänge. Sämtliche Ein- und Ausgänge sind TTL-kompatibel, auch der Einphasen-Takteingang. Die minimale Prozessorzykluszeit beträgt 2,4 µs. Besonderheiten sind der 15-bit-Adreßbus (32 KByte adressierbar), sieben 8-bit-Mehrzweck-Register, ein Unterprogramm-Rücksprungadreßspeicher mit 8 Ebenen, ein 14-bit-Programm-Status-Register (mit E/A-Leitungen gekoppelt) und vektorisierte Programm-Unterbrechung. Der Prozessor arbeitet mit 75 Grundbefehlen, die 1, 2 oder 3 Bytes lang sein können. Für die Befehle gibt es acht verschiedene Adressierungsarten, u. a. Indizierung mit wahlweisem Erniedrigen bzw. Erhöhen des Indexregisters. Der Datenaustausch mit Speichern und E/A-Ports kann syn-

Bild 1. Nicht gerade billig, aber immerhin mit Netzteil und Gehäuse – der Mikrocomputer Instructor-50 von Valvo/Signetics

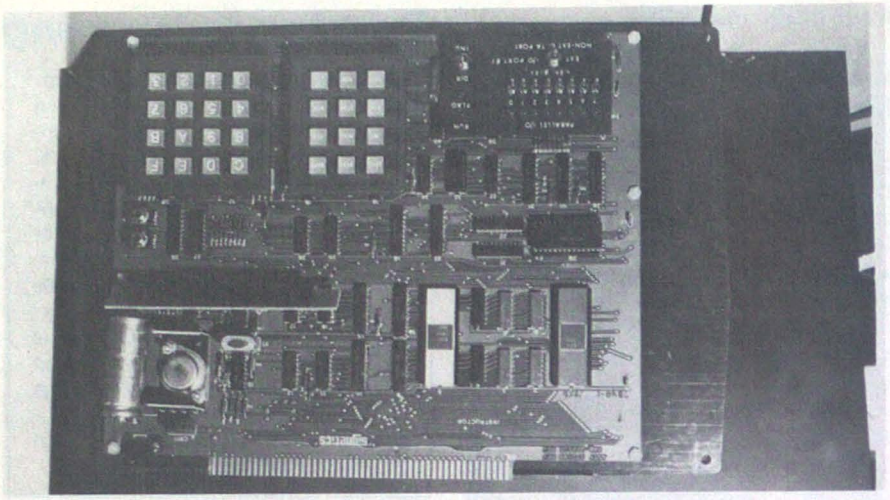


Bild 2. Der gesamte Mikrocomputer ist auf einer einzigen Pertinax-Platine untergebracht (Foto: Neumayr)

chron oder asynchron (quittungsgesteuert) abgewickelt werden. Die folgenden acht Adressierungsarten unterscheiden sich in der Art der Adreßberechnung im Speicherbedarf und in der Ausführungszeit:

- unmittelbare Angabe des Operanden,
- Registeradressierung,
- relative Adressierung,
- relative, indirekte Adressierung,
- absolute Adressierung,
- absolute, indirekte Adressierung,
- absolute Adressierung mit Indizierung,
- absolute, indirekte Adressierung mit Indizierung.

Die sieben Mehrzweckregister enthalten Operanden bzw. Ergebnisse bei arithmetischen und logischen Operationen und werden als Indexregister und für Ein-/Ausgabe-Vorgänge verwendet. Der Prozessor kann 32 768 Speicherplätze in 4 Seiten (pages) zu je 8192 Bytes adressieren. Ein direkter Speicherzugriff läßt sich auf einfache Weise z. B. durch Anhalten des Taktes realisieren.

Auf der Platine befinden sich ROMs mit insgesamt 2 KByte und RAMs mit 0,64 KByte. An der Rückseite des Gerätes ist der S-100-Bus herausgeführt, womit der Kontakt zur „übrigen Welt" hergestellt werden kann. *H. Neumayr*

Berichtigung Impedanzwandler-Berechnung

FUNKSCHAU 1979, Heft 25, Seite 1474

In das SR-56-Programm haben sich leider drei Druckfehler eingeschlichen, die dafür sorgen, daß eine „Iteration ins Ungewisse" erfolgt. An Adresse 13 muß statt „–" ein „:" stehen, an Adresse 41 statt „+" ein „x" und an Adresse 45 statt „–" ein „:".

µC-Nachrichten

Als Ergänzung der kurzen Industriemessen gibt es jetzt in Großbritannien eine Messe für die Elektroindustrie mit dauerhaftem Sitz im **Londoner World Centre.** Die Messe wird von der britischen Regierung unterstützt und zählt zu ihren Ausstellern solche Großunternehmungen wie IBM, GEC und Ferranti, sowie weniger bekannte kleinere Firmen. Der Kauf eines Standes ermöglicht die Benutzung aller Einrichtungen des Londoner World Trade Centres. Man erhält auch einen besonders für die jeweiligen Bedürfnisse entworfenen Stand (im Preis einbezogen). Die Messehalle wird von einer großen Zahl von Fachleuten besucht, d. h. von potentiellen Käufern die die neuesten technischen Entwicklungen sehen wollen.

□

Egil Juliussen von Texas Instruments äußerte diese Vision eines **Verkaufsgespräches der Zukunft:** Kauft unsere Speicher – und ihr erhaltet unsere Mikrocomputer umsonst; kauft unsere Peripherie – und ihr erhaltet unseren Minicomputer umsonst; kauft unsere Software – und ihr erhaltet unseren Standardrechner umsonst (aus Diebold Management Report, Okt. 1979).

□

Alfred Neye-Enatechnik bietet jetzt für die Mikrocomputer-Platine SDK-85 einen integrierten **Editor/Assembler** an, der AMSEDIT heißt und auch die symbolische Adressierung gestattet; maximal 255 Labels sind zulässig. Es handelt sich dabei um eine 80 x 80 mm² große Platine mit 4 KByte EPROM im Adressenbereich 8000...9FFF.

TI-59-Modul „Electrical Engineering"

Für zahlreiche Leser, die den programmierbaren Taschenrechner TI-59 besitzen, wird das von Texas Instruments herausgebrachte „Solid-State-Software"-Modul „Electrical Engineering" eine wertvolle Hilfe bei der Dimensionierung elektronischer Schaltungen darstellen. Wir stellen daher hier kurz die mit ihm möglichen Berechnungen vor. In vielen Fällen erstellt dabei, wenn angeschlossen, der Drucker PC-100 automatisch ein Protokoll von Eingabewerten und Ergebnissen.

Ausgesprochen schwierig ist oft die Berechnung von Phase Locked Loops (PLL-Schaltungen). Mit wenigen Tastendrücken erlaubt das Modul die Errechnung der Schleifen-Resonanzfrequenz, des Dämpfungsfaktors, der Filter-Bauelementewerte für aktive und passive Schleifenfilter sowie der Störbandbreite.

Ähnlich komfortabel ist der Umgang mit Polynomen; das Modul enthält Programme (die sich auch innerhalb eigener Routinen als Unterprogramme verwenden lassen) für das Herausfinden der Wurzeln und Multiplikationen bis zum 40. Grad. In der Wechselstromtechnik hat man oft mit komplexen Größen zu tun; dafür stehen komplexe Rechenarten und Funktionen zur Verfügung.

Für den Hf-Techniker ersetzt der Rechner ein Smith-Diagramm: Wieder genügen wenige Tastendrücke, um komplexe Leitungstransformationen zu errechnen. Wer einen TI-59 mit Drucker besitzt, kann sogar Netzwerk-Analysen durchführen. Hierbei wird der Frequenz- und Phasengang eines beliebigen linearen Netzwerks aus Widerständen, Kondensatoren, Induktivitäten und spannungsgesteuerten Stromquellen (z.B. Transistoren) errechnet. Das Netzwerk kann z.B. aus 5 Knotenpunkten und 9 Bauelementen oder aus 4 Knotenpunkten und 14 Bauelementen bestehen. Allerdings: Dafür braucht der Rechner schon ein wenig Zeit... Fe.

Rudolf Hofer

Schreibmaschine wird zum Schönschreibdrucker

Wenn Sie noch keinen Drucker haben oder mit der Qualität Ihres Druckers nicht zufrieden sind, sollten Sie die nächsten Seiten aufmerksam studieren. Hier wird eine preiswerte Alternative vorgestellt, die Ihnen eines Ihrer Computer-Probleme abnehmen könnte.

Über kurz oder lang kommt jeder Computer-Hobbyist an den Punkt, wo er schwarz auf weiß festhalten will, was seine Maschine von sich gibt. Meist ist dieser Zeitpunkt nicht allzu weit von der Anschaffung des Computers selbst entfernt, und deshalb kann der Geldbeutel nicht über Gebühr strapaziert werden. Als Lösung des Problems bieten sich preiswerte Matrixdrucker für einige hundert Mark an, die aber sehr begrenzte Möglichkeiten aufweisen (geringe Schreibbreite, Ansteuerung umständlich, Spezialpapier notwendig). Es gibt zwar auch komfortable Geräte, aber für sie muß man schon knapp 2000 Mark oder mehr hinblättern. Dennoch sind auch sie nicht der Weisheit letzter Schluß – für die Textverarbeitung (Briefe, Manuskripte und ähnliches) sind sie im allgemeinen nicht brauchbar.

Eine dritte Möglichkeit stellen elektrische Schreibmaschinen dar. Zwar liegen Kugelkopfmaschinen mit Dateneingang außerhalb der Reichweite von Hobbyisten (ab 5000 DM), doch hier soll die Rede von ganz gewöhnlichen Modellen sein, die um ein entsprechendes Interface erweitert werden. Die entstehenden Kosten liegen bei gut 400 DM (Bausatz).

Das Prinzip

Jede elektrische Schreibmaschine setzt die Bewegung einer Taste mechanisch in die Bewegung der Typenhebel bzw. des Kugelkopfes um. Man sucht also vergeblich nach einem elektrischen Kontakt, der durch die Taste geschlossen wird. Die einzige Möglichkeit, ein Zeichen auf ein elektrisches Signal hin zu drucken, besteht demnach darin, daß dieses in eine Bewegung umgesetzt wird. Man braucht also ein elektromechanisches Interface. Zu deutsch: Man braucht Magneten, von denen die Tasten bewegt werden.

In Kugelkopfmaschinen mit Dateneingang wird das im Prinzip genauso gemacht. Nur wird hier nicht jede Taste von einem Magnet betätigt, sondern durch eine Kombination mehrerer, die gleichzeitig anziehen. Dadurch kommt man mit wenigen Magneten aus. Diese Möglichkeit kommt für den Selbstbau nicht in Betracht, da man hierfür eine geeignete Maschine braucht und der Eingriff eine ganze Menge Spezialwissen erfordert.

Bild 1 zeigt das Prinzip des „Schreibmaschinen-Druckers". Auf dem Tastenfeld sitzt das elektromechanische Interface. Über jeder verwendeten Taste befindet sich eine Spule, die als Elektromagnet wirkt. Ein Stößel, der durch diese Spule hindurchragt, betätigt die Taste. Damit man mit möglichst wenigen Zuleitungen auskommt, sind die Magneten so in einer Matrix angeordnet, daß sie nur ansprechen, wenn gleichzeitig Zeile und Spalte freigegeben sind.

Zwischen der Magnetanordnung und dem Computer befindet sich eine Schaltung, die die parallel ankommenden Daten so decodiert, daß jeweils nur eine Zeile und eine Spalte aktiv sind, daß also zu jedem Zeitpunkt nur ein einziger Magnet ansprechen kann. Das gewählte Prinzip wurde aus mehreren Gründen einem Einbau der Magneten in die Maschine vorgezogen:

1. Bei den meisten Schreibmaschinen steht innen sehr wenig Platz zur Verfügung. Das ist um so kritischer, als die Tastenhebel nicht mehr zueinander versetzt, sondern in einer Reihe liegen.
2. Es ist kein Eingriff in die Maschine notwendig; eventuelle Garantieansprüche bleiben erhalten.
3. Die Abmessungen der Tastatur sind genormt (DIN), so daß ein Interface für beliebige Maschinen geeignet ist.

Praktische Ausführung

Das wichtigste Element des ganzen Aufbaus ist der Elektromagnet. Entscheidend ist die Kraft, die er aufbringen muß, um eine Taste zu betätigen. Dabei treten Unterschiede zwischen den Tasten derselben Maschine auf. Ein besonderes Problem stellen die schwergängigen Sondertasten wie Wagenrücklauf und Groß-/Klein-Umschaltung dar. Manchmal wird auch für die Leertaste etwas mehr Kraft benötigt. Alle „druckenden" Zeichen sind bei den meisten Maschinen so leicht zu betätigen, daß man mit ihnen keine Probleme hat.

Die größte Schwierigkeit beim Aufbau des Versuchsmodells bestand darin, daß keine geeigneten Spulen aufzutreiben waren. Also wurden sie per Hand gewickelt. Wenn geeignetes Werkzeug fehlt, ist dies eine recht mühsame Angelegenheit, vor allem dann, wenn man hinterher feststellt, daß viele nicht zu gebrauchen sind, weil einige Windungen kurzgeschlossen sind.

Dies war der Hauptgrund dafür, daß der Autor mit einer Firma in Verbindung getreten ist, die einen Bausatz mit

Innerhalb weniger Stunden läßt sich der komplette Interface-Bausatz aufbauen

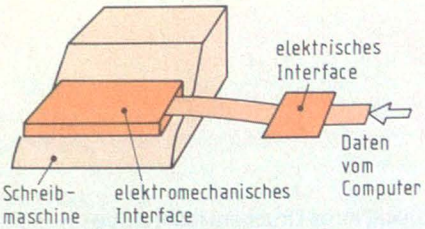


Bild 1. Die vom Computer kommenden Daten werden von einem elektrischen Interface so umgesetzt, daß man ein elektromechanisches Interface damit ansteuern kann

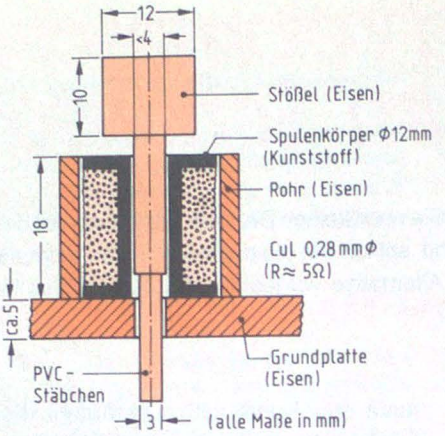


Bild 2. ▶

Mit diesem Magneten lassen sich die Tasten der meisten Schreibmaschinen betätigen. Für schwergängige Sondertasten sollte der Stößel mit seinem Eigengewicht eine gewisse „Vorspannung" erzeugen. Für Normaltasten genügt ein Eisenstift ohne Zusatzgewicht. Die optimale Eintauchtiefe im Ruhezustand ist ein wesentliches Kriterium für die erreichbare Kraft. Sie sollte mit einer Versuchsreihe ermittelt werden, bei der das PVC-Stäbchen verschieden lang gewählt wird

den geeigneten Magneten und dem gesamten mechanischen und elektrischen Aufbau herstellt.

Der Magnet

Der Idealmagnet, der auf jeden Fall alle „Zeichen"-Tasten betätigen kann, hat etwa die in *Bild 2* angegebenen Werte. Bei vielen Maschinen ist er sogar in der Lage, auch die kritischen Tasten zu bedienen. Sollte er für letztere bei 12 V nicht ausreichen, kann man durch eine höhere Spannung den Strom erhöhen. Diese Möglichkeit wird später noch näher erläutert. Wichtig für einen möglichst hohen Wirkungsgrad ist ein magnetischer Kreis, der am besten – bis auf einen „Luftspalt" – mit magnetisch gut leitendem Material geschlossen ist. Dies wird einerseits durch die Weicheisen-Grundplatte, andererseits durch das übergestülpte Rohr und den in die Spule eintauchenden Stößel (beides aus demselben Material) erreicht. Zu beachten ist, daß der Stößel nicht durchgehend aus Eisen sein darf, da er ja sonst keinen „Luftspalt" mehr freiläßt, den das magnetische Feld zu verringern sucht. Der untere Teil besteht deshalb aus einem PVC-Stäbchen, das man am besten etwas dünner wählt als den Stößel. Verbinden kann man die beiden Teile, indem man das PVC-Stäbchen in eine entsprechend große Bohrung des Stößels einpreßt. Kleben sollte man vermeiden, da sich schon geringe Reste des Klebstoffs recht störend auswirken können.

Bei den Sondertasten hat es sich als vorteilhaft erwiesen, wenn der Magnetstößel ein gewisses Eigengewicht (etwa 10 g) mitbringt. Es darf aber nicht zu hoch sein, sonst kann die Taste nicht in die Ruhestellung zurückkehren. In Bild 2 sind geeignete Maße angegeben.

Anordnung des Interface

In *Bild 3* ist ein Ausschnitt aus der Gesamtanordnung dargestellt. Man erkennt, daß auf den Magneten eine Leiterplatte aufliegt, auf der die notwendigen Dioden untergebracht sind. Außerdem sind die Spulen hier so verdrahtet, daß sich die erwähnte Matrixanordnung ergibt. An den vier Ecken der gesamten Einheit sind Kunststoffstäbe so angebracht, daß sie neben der Tastatur auf dem Schreibmaschinengehäuse aufsitzen. Um das Ganze zu fixieren, genügt ein Gummiband, das an beiden Enden des Interface befestigt und unten um den Rumpf der Maschine geschlungen wird. Eine ebenso primitive wie wirkungsvolle Methode ist es, zwei Einweckgummis ineinander zu verschlingen und die beiden Enden mit zu Ringen verlöteten Kupferdrähten an Schrauben einzuhängen, die eigens dafür vorgesehen sind. Das gesamte Interface läßt sich so mit zwei Handgriffen befestigen und wieder abnehmen (*Bild 4*). Perfektionisten werden aber sicher noch ganz andere Möglichkeiten finden.

Die Matrix

Wie schon erwähnt, werden die Magneten so angesteuert, daß jeder im

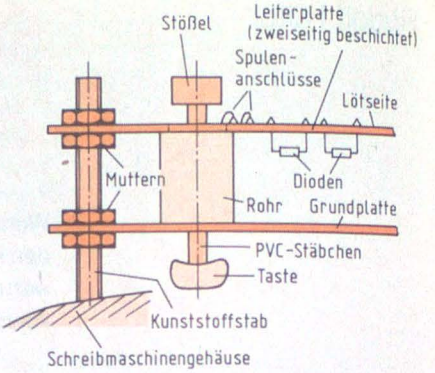


Bild 3. Ausschnitt aus der Gesamtanordnung: Auf der Leiterplatte sind die Magneten in einer Matrix verdrahtet und alle Dioden untergebracht; der Abstand zu den Tasten wird mit dem Kunststoffstab eingestellt

Kreuzungspunkt einer Zeilenleitung mit einer Spaltenleitung liegt. Insgesamt werden je sieben Zeilen und Spalten benötigt. Damit kann man 49 Funktionen abdecken. Das Prinzip zeigt *Bild 5*. Ein Magnet wird immer dann betätigt, wenn der zugehörige Spalten- und Zeilenschalter geschlossen sind. Die Schalter werden durch Transistoren realisiert, die auf dem elektrischen Interface (Bild 1) untergebracht sind. Damit keine unerwünschten Querströme auftreten, liegt in jeder Spulenzuleitung eine Diode, die den ganzen „Arbeitsstrom" aushalten muß (1N4001). Eine weitere Diode (1N4148) dient zum Schutz der Transistoren wegen der induktiven Last. Alle Dioden – insgesamt knapp 100 Stück – sitzen auf der Verdrahtungsplatine des elektromechanischen Interface.

Eine große Schwierigkeit lag darin, die quadratische 7 x 7-Matrix auf die rechteckige Anordnung der Schreibmaschinentastatur zu übertragen. Einen brauchbaren Kompromiß, der wenigstens eine einigermaßen regelmäßige Struktur erkennen läßt, stellt *Bild 6* dar. Absichtlich wurden hier alle möglicherweise schwergängigen Tasten unter einer Spalte zusammengefaßt. Im Bedarfsfall kann man dann

So wird das Interface auf die Schreibmaschine aufgesetzt

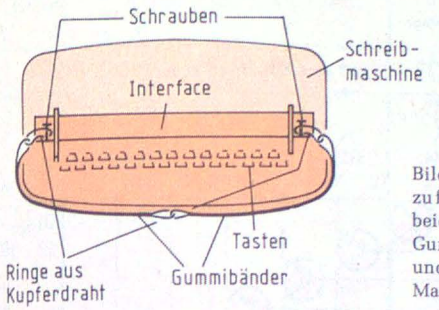


Bild 4. Um das Interface zu fixieren, kann man an beiden Seiten ein Gummiband befestigen und um den Rumpf der Maschine schlingen

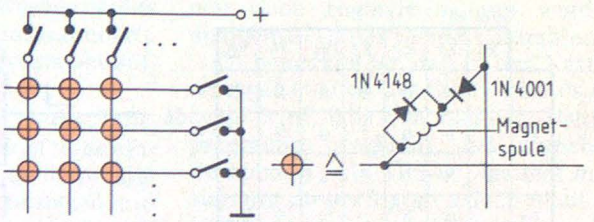


Bild 5. Nur wenn ein Zeilen- und ein Spaltenschalter geschlossen sind, wird der Magnet im Schnittpunkt von Strom durchflossen

nämlich die Spannung an dieser einen Spalte so weit erhöhen, daß genügend Kraft zur Verfügung steht.

Aus Bild 6 ergibt sich unmittelbar das Schema von *Bild 7*, und daraus leitet sich wiederum der Code ab, mit dem das elektrische Interface angesteuert wird.

Das elektrische Interface

Bild 8 zeigt die Schaltung des elektrischen Interface. Dessen Aufgabe besteht darin, den vom Computer kommenden parallelen Code (D 0...D 5) so zu decodieren, daß jeweils nur ein Spalten- und Zeilentransistor durchgeschaltet werden. Man erreicht das mit zwei BCD/Dezimal-Decodierern, an deren Eingängen das Datenwort anliegt. Die Ausgänge des Typs 7445 sind für eine maximale Spannung von 30 V und einen maximalen Strom von 80 mA ausgelegt. Im Versuchsaufbau wurden Darlingtontransistoren verwendet, damit die Stromverstärkung vor allem der NPN-Typen, die mit normalen TTL-Invertierern angesteuert werden, ausreicht. Man könnte aber auch das preiswertere Transistorenpaar BD135/136 verwenden und die Ansteuerung entsprechend ändern (statt 7404 ein Typ mit offenem Kollektor und Widerständen nach $U_{Magn.}$; Vorwiderstände der PNP-Typen kleiner). Für schwergängige Sondertasten kann man die Versorgungsspannung des entsprechenden PNP-Typs abtrennen und soweit erhöhen, bis auch sie einwandfrei betätigt werden. Wie erwähnt, wurden alle in Frage kommenden Funktionen aus diesem Grund auf dieselbe Spalte gelegt.

Üblicherweise zeigt der Computer mit einem Pegelwechsel des Strobe-Signals an, daß die angelegten Daten gültig sind und das entsprechende Zeichen ausgedruckt werden soll. Dieses Strobe-Signal wird durch ein Monoflop verlängert und den D-Eingängen der Decodierer zugeführt. Sind diese Eingänge H, dann sind alle verwendeten Ausgänge ebenfalls auf H (nicht aktiv), egal was an den Dateneingängen A...C anliegt. Die D-Eingänge wirken also wie Aktivierungseingänge. Kommt ein Strobe-Impuls an, werden für kurze Zeit die Decodierer aktiviert, und die Ausgänge steuern genau für diese Zeit einen Spalten- und Zeilentransistor an. Damit wird verhindert, daß versehentlich eine Magnetspule dauernd von Strom durchflossen wird, wenn am Strobe-Eingang ein falscher Dauerpegel anliegt.

Der Monoflop-Ausgang wird gleichzeitig als Rückmeldung für den Computer verwendet, ob der Drucker empfangsbereit ist oder nicht (Busy). Damit läßt sich ein Quittungsbetrieb (*Handshake*) realisieren, wie er allgemein üblich ist.

Die Zeitkonstante des Monoflops – sie bestimmt die Anzugsdauer des Magneten – muß experimentell ermittelt werden. In der Praxis liegt dieser Wert beim 50 ms. Das zusätzliche Monoflop und der „Shift"-Transistor sind nicht unbedingt erforderlich. Man kann darauf verzichten, wenn man die später beschriebene Art der Groß-/Klein-Umschaltung wählt.

Der Code

Der Grund dafür, warum man das Interface nicht im ASCII-Code ansteuern kann, liegt darin, daß manche Zeichen bei der Schreibmaschine „Shift" erfordern, während sie im ASCII-Code ohne Shift dargestellt werden und umgekehrt.

Man muß deshalb eine Codierung vornehmen, an der eindeutig erkennbar ist, ob es sich um ein Zeichen handelt, das mit oder ohne Betätigung der Shift-Taste gedruckt wird. Der Code muß außerdem so ausgelegt sein, daß jedes Zeichen dem bereits festgelegten Matrixschnittpunkt von Bild 7 entspricht. *Bild 9* zeigt die Zuordnung von

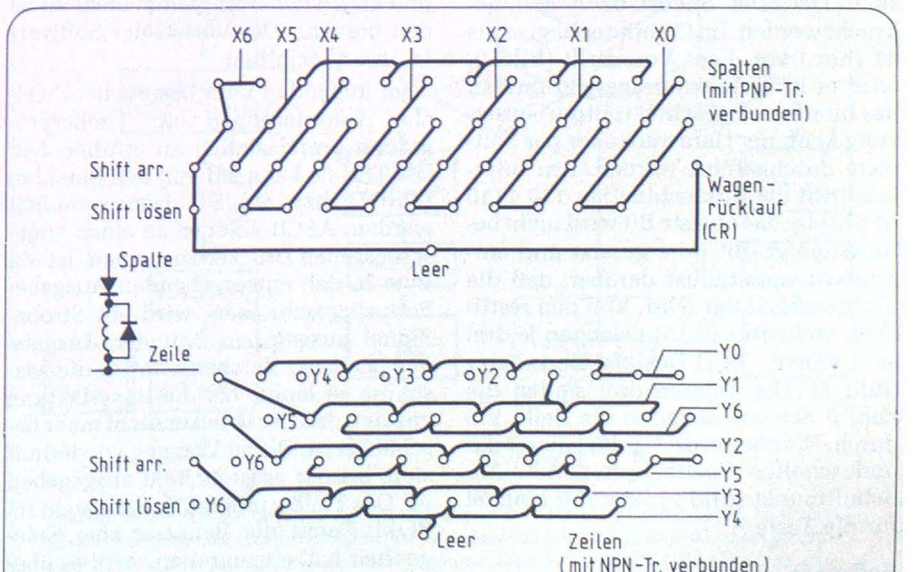


Bild 6. Zuordnung der Tasten zu Spalten und Zeilen: Die Verdrahtung wird auf der in Bild 3 erkennbaren Leiterplatte vorgenommen

Zeilen \ Spalten	X0	X1	X2	X3	X4	X5	X6
Y0	'	+	8	6	4	ß	2
Y1	ü	o	u	t	e	9	1
Y2	ö	k	h	f	s	7	9
Y3	.	m	b	c	y	5	Sha.
Y4	–	,	n	v	x	3	BL*
Y5	ä	l	j	g	d	w	CR*
Y6	←*	p	i	z	r	a	Sh.L.

* BL = Leer; CR = Wagenrücklauf;
← = Rücktaste

Bild 7. Diese Zuordnung der Tasten ergibt sich aus Bild 6 (für Gabriele 2000, Triumph/Adler)

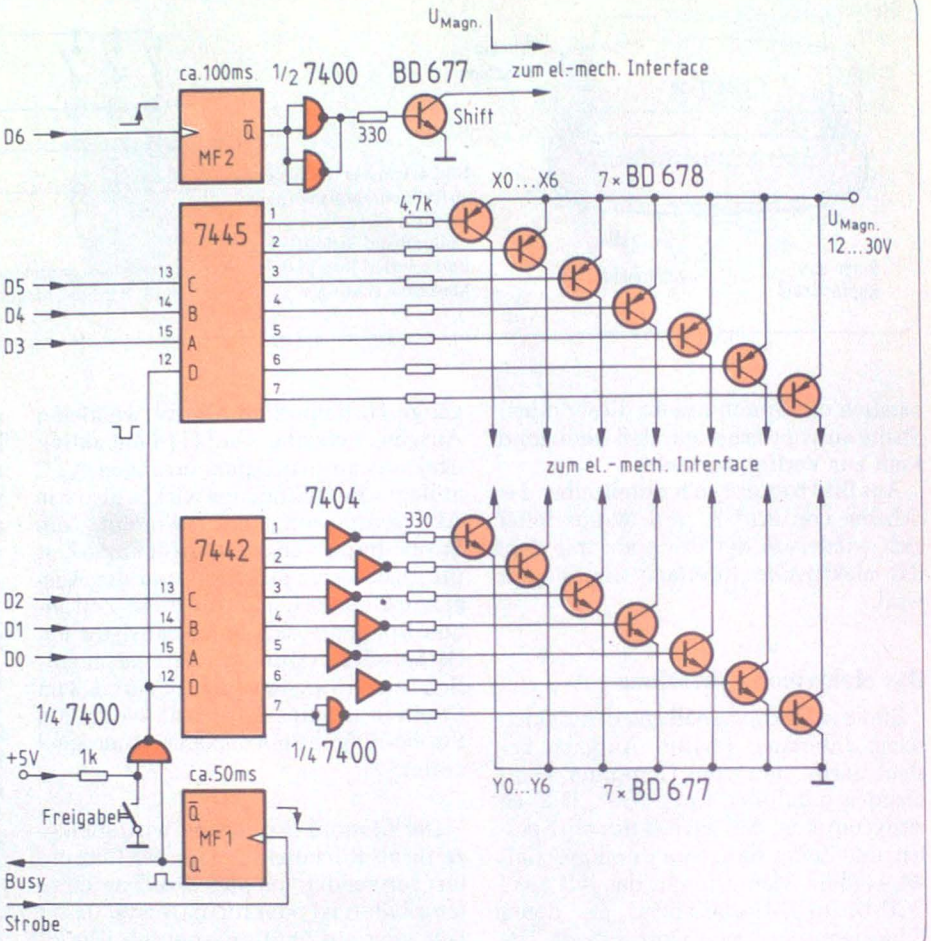


Bild 8. ▶
Schaltung des elektrischen Interface: Die Shift-Taste muß nicht gesondert verdrahtet werden. Man kann D6 auch weglassen und die Groß/Klein-Umschaltung mit Hilfe der Tasten „Shift-Arretierung" und „Shift lösen" vornehmen. In diesem Fall erhalten die beiden Funktionen je eine Bitkombination der Eingänge D 0...D 5 zugewiesen

ASCII-Code und Drucker-Code. Das Erkennungsmerkmal für ein Zeichen mit Shift ist, daß Bit 6 (im Drucker-Code) gesetzt ist. Am besten macht man sich den Weg eines Zeichens an einem Beispiel klar: Der Buchstabe A soll gedruckt werden. Im Computer liegt er als 41 (hex.) vor. Laut Vorschrift (Bild 9) wird er in 6E (hex.) umgesetzt und an das Interface abgeschickt. Diese Umsetzung kann per Hardware oder per Software durchgeführt werden. Am Interface trifft die Bitkombination 110 1110 (≙ 6E) ein; das oberste Bit wird nicht berücksichtigt. Bit 6 ist gesetzt und entscheidet unmittelbar darüber, daß die Shifttaste betätigt wird. Von den restlichen sechs Bits (0...5) gelangen je drei an einen BCD/Dezimal-Decodierer (Bild 8). Die unteren drei stellen die Zahl 6 dar und schalten die Zeile Y 6 durch. Die oberen drei stellen eine 5 dar und schalten Spalte X 5 durch. Am Schnittpunkt (Bild 7) liegt der Magnet für die Taste A.

Software

Wie angedeutet, kann man die Code-Umsetzung von zusätzlicher Hardware oder von Software durchführen lassen. Die Hardware-Lösung hat den Vorteil, daß man die Schreibmaschine mit dem Interface dann wie einen Standarddrucker anschließen kann und kein Treiberprogramm mehr zu laden braucht. Der Vorteil der Software ist ihre Flexibilität.

Im folgenden wird besprochen, welche Anforderungen ein Treiberprogramm grundsätzlich zu erfüllen hat. Der TRS-80 kann z.B. mit den Befehlen LPRINT bzw. LLIST dazu veranlaßt werden, ASCII-Zeichen an einen angeschlossenen Drucker zu senden. Ist ein Byte in den entsprechenden Ausgabe-Port eingeschrieben, wird ein Strobe-Signal ausgegeben. Mit der Ausgabe des nächsten Zeichens wartet die Maschine so lange, bis die Busy-Leitung anzeigt, daß der Drucker nicht mehr beschäftigt ist. Dieser Vorgang wiederholt sich, bis der gesamte Text ausgegeben ist. Das Treiberprogramm dazu steht im ROM. Damit der Benutzer aber Gelegenheit hat, einzugreifen, wird es über eine RAM-Zelle angesprungen. D. h. bei jedem Druckbefehl holt sich der Computer die Anfangsadresse des Treiberprogramms aus dieser RAM-Zelle. Ändert man ihren Inhalt, dann springt der Computer z. B. auf das Treiberprogramm des Benutzers. Normalerweise legt man den eigenen Treiber an das Ende des RAM-Bereichs, der dafür reserviert wird. Um die ganzen Organisationsprobleme (Zeilenende erreicht?, Steuerzeichen? usw.) zu umgehen, kann man folgendermaßen vorgehen:

– Der ursprüngliche Treiber im ROM wird in einen RAM-Bereich verlegt.
– Die Sprungadresse wird so geändert, daß sie auf den neuen Anfang zeigt.
– Alle Schreiboperationen zum Port, an dem der Drucker hängt, werden durch einen Unterprogramm-Aufruf ersetzt.

Das Unterprogramm (*Bild 10*) ist der einzige Teil, den man selber schreiben muß. Er hat im wesentlichen die Aufgabe, den ASCII-Code, der ursprünglich zum Drucker geschickt werden sollte, durch den Drucker-Code für die Schreibmaschine zu ersetzen und den dann auszugeben.

Wie aus dem Flußdiagramm ersichtlich, beginnt das Programm mit einer Verzögerung von 50 ms (die Zeitangaben können je nach Schreibmaschine

Zeichen gr.	Zeichen kl.	Drucker (hex.) gr.	Drucker (hex.) kl.	ASCII (hex.) gr.	ASCII (hex.) kl.
`	´	40	00	–	–
Ü	ü	41	01	–	–
Ö	ö	42	02	–	–
!	.	43	03	21	2E
'	–	44	04	27	2D
Ä	ä	45	05	–	–
←	←	46	06	08	08
/	+	48	08	2F	2B
O	o	49	09	4F	6F
K	k	4A	0A	4B	6B
M	m	4B	0B	4D	6D
?	,	4C	0C	3F	2C
L	l	4D	0D	4C	6C
P	p	4E	0E	50	70
–	8	50	10	–	38
U	u	51	11	55	75
H	h	52	12	48	68
B	b	53	13	42	62
N	n	54	14	4E	6E
J	j	55	15	4A	6A
I	i	56	16	49	69
(	6	58	18	28	36
T	t	59	19	54	74
F	f	5A	1A	46	66
C	c	5B	1B	43	63
V	v	5C	1C	56	76
G	g	5D	1D	47	67
Z	z	5E	1E	5A	7A
%	4	60	20	25	34
E	e	61	21	45	65
S	s	62	22	53	73
Y	y	63	23	59	79
X	x	64	24	58	78
D	d	65	25	44	64
R	r	66	26	52	72
:	ß	68	28	3A	–
§	9	69	29	–	39
)	7	6A	2A	29	37
&	5	6B	2B	26	35
=	3	6C	2C	3D	33
W	w	6D	2D	57	77
A	a	6E	2E	41	61
"	2	70	30	22	32
;	1	71	31	3B	31
Q	q	72	32	51	71
Sh.a.	Sh.a.	73	33	–	–
BL	BL	74	34	20	20
CR	CR	75	35	0D	0D
Sh.l.	Sh.l.	76	36	–	–

Bild 9. Drucker-Code (für Gabriele 2000) und die entsprechenden ASCII-Verschlüsselungen: Diese Tabelle ist die Grundlage für die Code-Umsetzung, die das Treiberprogramm ausführen muß. Für Zeichen ohne ASCII-Äquivalent kann man ähnlich aussehende ASCII-Zeichen verwenden, z. B. $ für §, a für ä, # für ü, Ø für ö. Umgekehrt weist man der Null das „O" zu und dem Malzeichen das „x"

variieren). Das ist die Beruhigungszeit für den vorher betätigten Magneten. Es folgt die Code-Wandlung. Man nimmt sie am besten mit einer Tabelle entsprechend Bild 9 vor. Änderungen kann man damit am bequemsten durchführen.

Falls es sich beim auszugebenden Zeichen um Wagenrücklauf handelt, wird der Code dafür ausgegeben, und nach einer Verzögerung von ca. 0,5 s springt der Computer in das Hauptprogramm zurück. Liegt ein anderes Zeichen vor (Steuerzeichen werden bereits vom Hauptprogramm ausgesondert), wird ein weiteres Unterprogramm aufgerufen.

Hier wird entschieden, ob die Shift-Taste betätigt wird oder nicht. Eine einfache Möglichkeit bestünde darin, alle Bits außer Bit 6 auf Null zu setzen und das Byte auszugeben, falls es nicht null ist. Damit würde man die Shift-Taste betätigen. Nach einer gewissen Zeit (z.B 50 ms) könnte man den eigentlichen Code hinterherschicken und würde damit das gewünschte Zeichen anschlagen. Der Nachteil ist offensichtlich: Bei jedem Großbuchstaben wird die Shift-Taste betätigt und wieder losgelassen.

Besser ist es, die im Flußdiagramm dargestellte Methode anzuwenden. Sie benutzt die Shift-Arretierungs-Taste und deren Gegenstück (Shift lösen). Damit wird nur umgeschaltet, wenn ein Wechsel von Groß auf Klein oder umgekehrt auftritt. Der Computer merkt sich den augenblicklichen Modus mit Hilfe einer RAM-Zelle, Großflag genannt. Die Verzögerungszeiten am Ende jeden Zweiges müssen so eingestellt werden, daß sich der Wagen bzw. Kugelkopf mit Sicherheit beruhigt hat. Verfährt man nach dieser Methode, dann kann man auf die gesonderte Ansteuerung der Shift-Taste verzichten. Die beiden Umschalter „Shift arret." und „Shift lösen" erhalten einfach einen Code zugewiesen und werden demgemäß in der Matrix verdrahtet.

Zu bemerken ist, daß in den Kästen „Ausgabe" auch der Strobe-Impuls erzeugt wird, falls dies nicht im Hauptprogramm geschieht. Bei manchen Computern (z.B. TRS-80) braucht man sich um dieses Signal jedoch nicht zu kümmern, da es vom Schreibsignal für den Ausgabe-Port hardwaremäßig abgeleitet wird. In ähnlicher Weise kann man Treiber von Textverarbeitungsprogrammen modifizieren, so daß man einen Schreibautomaten mit jeglichem Komfort erhält.

Erfahrungen

Der Autor hat das Interface an einer sehr preiswerten Maschine (Triumph/Adler Gabriele 2000, Ladenpreis etwa 500 DM) getestet. Die maximale Geschwindigkeit lag bei ca. 10 Anschlägen/s. Für einen sicheren Betrieb sollte man sich aber auf etwa 7 Anschläge/s beschränken. Mit Kugelkopfmaschinen läßt sich die Geschwindigkeit noch erheblich steigern. Das erwähnte Modell wurde ausgewählt, weil es besonders leichtgängige Tasten hat. Darauf sollte man achten, wenn man eine Maschine kauft.

Ein kompletter Bausatz ist vom Autor erhältlich (Ismaninger Straße 156, 8000 München 80). Passende Treiberprogramme für gängige CPU-Typen folgen in den nächsten Heften.

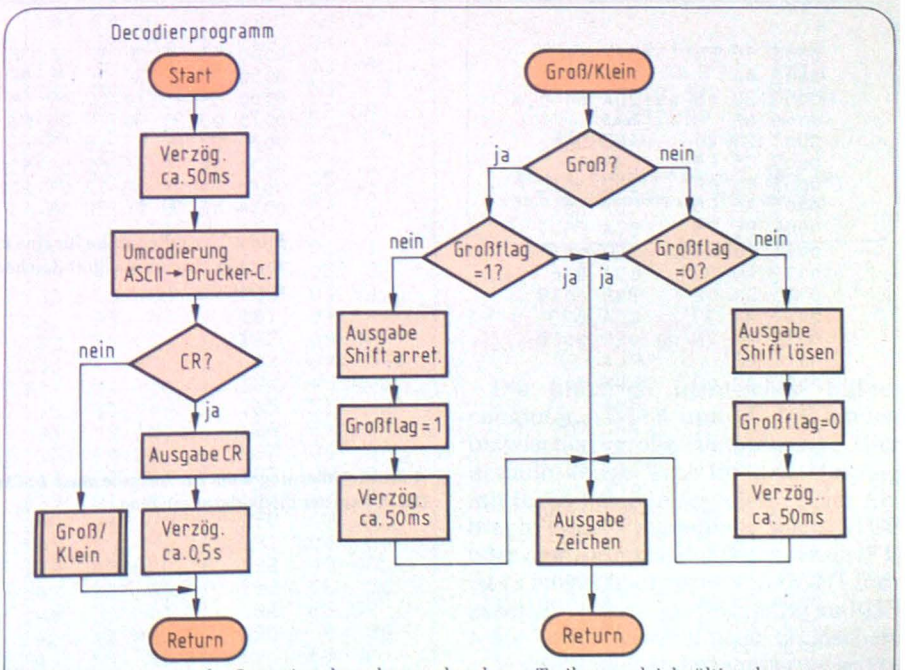


Bild 10. Jede Ausgabe-Operation des schon vorhandenen Treibers – gleichgültig, ob er zum BASIC-Interpreter, zu einem Textverarbeitungsprogramm oder zu einem Assembler gehört – wird durch einen Sprung auf dieses Unterprogramm ersetzt. Statt Groß/Klein kann man auch eine Routine verwenden, mit der man erst die Shifttaste betätigt und ca. 50 ms später den Zeichen-Code folgen läßt. Empfehlenswert ist das, wenn die Shift-Arretierungs-Taste sehr schwergängig ist.

6502-Routine zur Schreibmaschinen-Ansteuerung

Anhand eines praktisch ausgeführten Beispiels wird deutlich, wie einfach die Software für die Ansteuerung des auf Seite 85 beschriebenen Schreibmaschinen-Interface ist. Die in *Bild 1* disassemblierte 6502-Routine gestattet sogar noch eine Vereinfachung der Interface-Hardware: Das Monoflop 74123 kann ersatzlos entfallen. Die Leitung zwischen den Nand-Gatter und den D-Eingängen der ICs 7442 und 75145 ist aufzutrennen, letztere können direkt mit dem Anschluß PB7 des Mikrocomputers verbunden werden. (In der Praxis führte das Monoflop ohnehin mehrmals zu Problemen, weil es offenbar auch durch kurze und leider manchmal unvermeidliche Störimpulse getriggert wird.)

Die Ansteuerung der Interface-Leitungen D0...D5 erfolgt über die Mikrocomputer-Ausgänge PB0...PB5. Der schon erwähnte PB7 gibt den Magnetstromkreis frei, solange er auf Low-Pegel liegt. PB6 bleibt unbenutzt, und zusätzlich sind nur noch Leitungen für Masse, + 5 V und + 18...26 V (Magnetspannung) erforderlich. Alle Zeitkonstanten werden vom Programm bestimmt:

Adresse	*Inhalt*
001E	Dauer des Tastendrucks
0026	Kurze Erholzeit
002B	Lange Erholzeit

Die kurze Erholzeit wird bei allen Buchstaben- und Zifferntasten verwendet. Nur die Tasten „Return" (Wagenrücklauf) und „Backspace" (Rücktaste) sind in der Umcodiertabelle von *Bild 2* durch eine Eins im höchstwertigen Bit gekennzeichnet und werden mit der längeren Erholzeit ausgegeben, um den größeren Zeitbedarf der Mechanik bei diesen Tasten zu berücksichtigen.

Die Shift-Funktion wird hier durch die Shift-Arretiertaste (Code: Adresse 0018) und die Shift-Taste (Adresse 0014) ausgeführt. Die Adressenbelegung entspricht dem Mikrocomputer KIM-1, bei dem das Port-A-Register bei 1700 und das Port-A-Richtungsregister bei 1701 liegt. Da die Routine in Bild 1 nur ein ASCII-Zeichen ausdruckt, zeigt *Bild 3* ein kleines Testprogramm, mit dem man vom ASCII-Terminal aus auf der Schreibmaschine schreiben kann. Es holt ein Zeichen von der Terminal-Tastatur, schaltet alle Leitungen von Port A als Ausgang und gibt das Zeichen umcodiert an das Interface aus.

Die Codierung in Bild 3 entspricht einer deutschen Schreibmaschine („Gabriele" von Triumph-Adler). Das Bit 6 der Tabellenbytes gibt an, ob es sich um ein Shift-Zeichen handelt. Das Unterprogramm selbst läßt sich in jeden beliebigen Speicherbereich transferieren, wenn man alle dritten Bytes bei Drei-Byte-Befehlen von 00 auf die neue Page ändert. Da keine Timer-Funktionen verwendet werden, ist eine Änderung auf andere Mikrocomputer (AIM-65, SYM-1, PET-2001 usw.) mit keinerlei Problemen verbunden. Ein kleiner Programmiertrick findet sich bei 0033: Hier wird ein Unterprogrammsprung mit anschließendem Rücksprung dafür verwendet, eine Verzögerung um 12 Maschinenzyklen zu erreichen. Fe.

```
0000 86 F4     STX F4
0002 AA        TAX
0003 BD 40 00  LDA 0040,X
0006 48        PHA
0007 29 40     AND #40
0009 C5 F5     CMP F5
000B F0 0F     BEQ 001C
000D A6 F5     LDX F5
000F 85 F5     STA F5
0011 F0 04     BEQ 0017
0013 A9 36     LDA #36
0015 D0 02     BNE 0019
0017 A9 33     LDA #33
0019 20 1D 00  JSR 001D
001C 68        PLA
001D A2 08     LDX #08
001F 48        PHA
0020 29 3F     AND #3F
0022 20 2E 00  JSR 002E
0025 A2 10     LDX #10
0027 68        PLA
0028 10 02     BPL 002C
002A A2 30     LDX #30
002C A9 FF     LDA #FF
002E A0 00     LDY #00
0030 8D 00 17  STA 1700
0033 20 3E 00  JSR 003E
0036 88        DEY
0037 D0 F7     BNE 0030
0039 CA        DEX
003A D0 F4     BNE 0030
003C A6 F4     LDX F4
003E 60        RTS
```

```
0040 3F 3F 3F 3F 3F 3F 3F 3F C6 34 3F 3F 3F B5 3F 3F
0050 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F
0060 34 43 70 6C 69 60 6B 44 58 6A 24 08 0C 04 02 48
0070 09 31 30 2C 20 2B 18 2A 10 29 68 71 0A 6C 1D 4C
0080 2E 6E 53 5B 65 61 5A 5D 52 56 55 4A 4D 4B 54 49
0090 4E 72 66 62 59 51 5C 6D 64 63 5E 45 42 34 40 28
00A0 34 2E 13 1B 25 21 1A 1D 12 16 15 0A 0D 0B 14 09
00B0 0E 32 26 22 19 11 1C 2D 24 23 1E 05 02 34 01 46
```

Bild 2. Umcodiertabelle für eine deutsche Schreibmaschinentastatur. Bit 6 gibt an, ob es sich um ein Shift-Zeichen handelt, und Bit 7, ob eine längere Erholzeit notwendig ist

◄ Bild 1. Unterprogramm zur Ausgabe eines ASCII-Zeichens auf der Schreibmaschine. Das X-Register bleibt dabei erhalten

Bild 3. ► Einfaches Testprogramm zum Schreiben auf der Maschine vom ASCII-Terminal aus; Startadresse ist 00C0

```
00C0 20 5A 1E  JSR 1E5A
00C3 A0 FF     LDY #FF
00C5 8C 01 17  STY 1701
00C8 20 00 00  JSR 0000
00CB 4C C0 00  JMP 00C0
```

TI-59 plottet Funktionen

Das nachfolgend beschriebene TI-59-Programm gibt Datenpunkte auf dem Drucker PC-100 als Graph aus. Dabei wird automatisch der Maßstab so gewählt, daß der vorhandene Platz von 20 Zeichen pro Zeile optimal ausgenutzt wird.

In der in *Bild 1* aufgelisteten Programmversion sind maximal 50 Dateneingaben zulässig. Werden mehr gebraucht, so läßt sich dies leicht verwirklichen, indem man die Speicherbereichsverteilung z. B. auf 239.89 ändert, die Adressen der Hinweisregister (50...54) z. B. auf 85...89 verschiebt und die Schritte 20 und 21 ändert.

Der Gebrauch ist recht einfach: Nach dem Eingeben des Programms drückt man E; damit ist der Rechner vorbereitet und die Dateneingabe kann beginnen. Hierzu wird der Wert eingegeben und die Taste A gedrückt; danach wird die Anzahl der bisher eingegebenen Datenpunkte angezeigt. Sind alle Werte gespeichert, so wird nach Betätigen der Taste B der Graph ausgedruckt. Ferner werden der kleinste und größte Wert sowie der Mittelwert angegeben (*Bild 2*).

Rainer Maaß

```
000  76 LBL   050  55  55   100  54  54   150  24  24
001  15  E    051  01   1   101  00   0   151  43 RCL
002  47 CMS   052  22 INV   102  42 STO   152  55  55
003  22 INV   053  44 SUM   103  50  50   153  22 INV
004  86 STF   054  50  50   104  04   4   154  49 PRD
005  01  01   055  43 RCL   105  07   7   155  56  56
006  91 R/S   056  52  52   106  04   4   156  69 OP
007  76 LBL   057  32 X:T   107  07   7   157  05  05
008  11  A    058  73 RC*   108  04   4   158  69 OP
009  87 IFF   059  50  50   109  07   7   159  00  00
010  01  01   060  22 INV   110  04   4   160  04   4
011  23 LNX   061  77  GE   111  07   7   161  05   5
012  72 ST*   062  00  00   112  04   4   162  03   3
013  50  50   063  66  66   113  07   7   163  00   0
014  01   1   064  42 STO   114  69 OP    164  02   2
015  44 SUM   065  52  52   115  01  01   165  04   4
016  50  50   066  43 RCL   116  69 OP    166  03   3
017  43 RCL   067  53  53   117  02  02   167  01   1
018  50  50   068  32 X:T   118  69 OP    168  69 OP
019  32 X:T   069  73 RC*   119  03  03   169  04  04
020  04   4   070  50  50   120  69 OP    170  43 RCL
021  09   9   071  77  GE   121  04  04   171  53  53
022  22 INV   072  00  00   122  69 OP    172  69 OP
023  77  GE   073  76  76   123  05  05   173  06  06
024  23 LNX   074  42 STO   124  73 RC*   174  04   4
025  43 RCL   075  53  53   125  50  50   175  05   5
026  50  50   076  01   1   126  44 SUM   176  03   3
027  91 R/S   077  22 INV   127  56  56   177  00   0
028  76 LBL   078  44 SUM   128  75   -   178  01   1
029  23 LNX   079  50  50   129  43 RCL   179  03   3
030  86 STF   080  43 RCL   130  53  53   180  04   4
031  01  01   081  50  50   131  95   =   181  04   4
032  43 RCL   082  32 X:T   132  55   ÷   182  69 OP
033  50  50   083  01   1   133  43 RCL   183  04  04
034  94 +/-   084  94 +/-   134  54  54   184  43 RCL
035  34 √x    085  22 INV   135  95   =   185  52  52
036  33 x²    086  67  EQ   136  59 INT   186  69 OP
037  91 R/S   087  00  00   137  69 OP    187  06  06
038  76 LBL   088  55  55   138  07  07   188  06   6
039  12  B    089  43 RCL   139  01   1   189  07   7
040  25 CLR   090  52  52   140  44 SUM   190  69 OP
041  43 RCL   091  75   -   141  50  50   191  04  04
042  00  00   092  43 RCL   142  43 RCL   192  43 RCL
043  42 STO   093  53  53   143  50  50   193  56  56
044  52  52   094  95   =   144  32 X:T   194  69 OP
045  42 STO   095  55   ÷   145  43 RCL   195  06  06
046  53  53   096  01   1   146  55  55   196  69 OP
047  43 RCL   097  09   9   147  22 INV   197  00  00
048  50  50   098  95   =   148  67  EQ   198  98 ADV
049  42 STO   099  42 STO   149  01  01   199  98 ADV
                                          200  98 ADV
                                          201  91 R/S
```

Bild 1. Programmauflistung zum Darstellen von Datenpunkten auf dem Drucker des TI-59

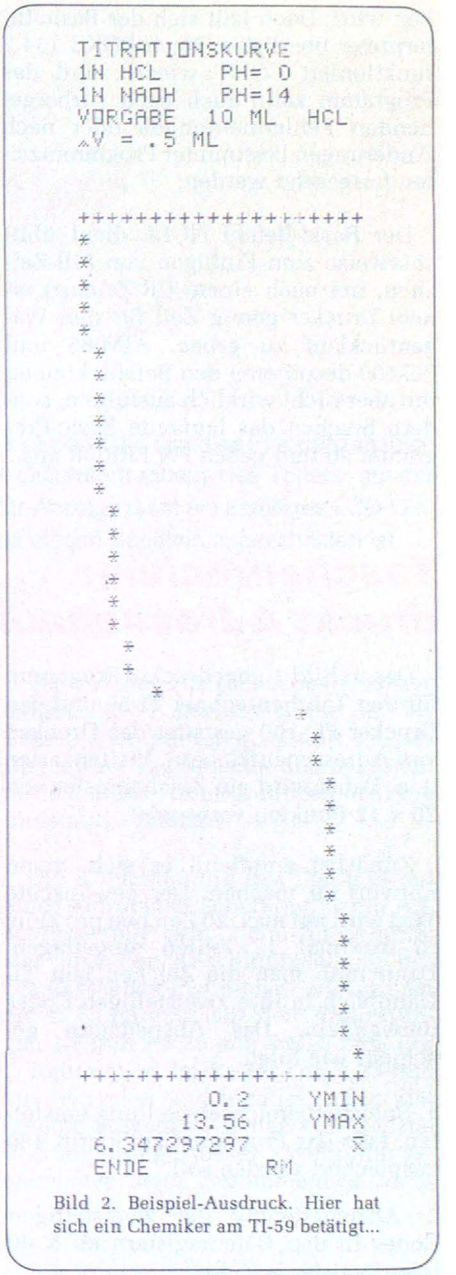


Bild 2. Beispiel-Ausdruck. Hier hat sich ein Chemiker am TI-59 betätigt...

AIM-65-Tips

Die praktisch identischen Mikrocomputer AIM-65 und PC-100 fanden inzwischen große Verbreitung. Hier deshalb einige Tips für den Umgang mit ihren Basic-Interpretern. Beim Abbruch eines Programmes nach STOP oder dem Drücken der Break-Taste (F1) ist es möglich, es später mit CONT fortzusetzen, ohne – im Gegensatz zu RUN – die Variablenwerte dabei zu löschen.

Vertippt man sich dummerweise vor dem CONT-Befehl, so handelt man sich zunächst einen SN ERROR (Syntaxfehler) ein, nach dem sich der CONT-Befehl nicht mehr anwenden läßt, weil nach CONT sofort CN ERROR ausgege-

ben wird. Doch läßt sich der Basic-Interpreter überlisten: Nach POKE 134,2 funktioniert CONT wieder, und das Programm kann auch nach vorhergehenden Fehlermeldungen oder nach Änderungen bestimmter Programmzeilen fortgesetzt werden.

Der Basic-Befehl NULL dient üblicherweise zum Einfügen von Fill-Zeichen, um nach einem CR-Zeichen einem Drucker genug Zeit für den Wagenrücklauf zu geben. AIM-65 und PC-100 decodieren den Befehl, können ihn aber nicht wirklich ausführen, sondern brechen das laufende Basic-Programm ab und geben FN ERROR aus.

Wie man in einem Basic-Programm Variablen, d.h. Texte und Zahlenwerte auf Kassette abspeichern und wieder laden kann, war bereits in Heft 1/1980 unter dem Titel „Basic-Texteditor" beschrieben. Es ist sogar möglich, mit dem AIM-Texteditor (im Monitor-ROM) Texte zu editieren, auf Kassette zu bringen und von Basic her zu laden. Aber: Die Texte dürfen keine Kommas und Doppelpunkte enthalten, da diese beiden Zeichen zur Trennung von Variablen dienen. Aus diesem Grund ist es leider (!) nicht möglich, ein komplettes Basic-Programm in String-Variablen einzulesen und per Basic-Editor zu editieren. Fe.

Taschenrechner druckt Adressenaufkleber

Das in *Bild 1* abgedruckte Programm für den Taschenrechner TI-59 und den Drucker PC-100 gestattet das Drucken von Adressenaufklebern, Visitenkarten u. a. Dabei wird ein Zeichenraster von 20 x 12 Punkten verwendet.

Zunächst empfiehlt es sich, einen Entwurf zu machen. Der gewünschte Text wird mit max. 20 Zeichen pro Zeile in maximal 12 Zeilen eingetragen. Dann muß man die Zeichen laut TI-Handbuch in ihre zweistelligen Codes umwandeln. Das Abspeichern geschieht wie folgt:

1. Speicherbereichseinstellung einstellen, falls das Programm ab Schritt 480 gespeichert werden soll.

2. Abspeicherung der zweistelligen Codes in den Datenregistern ab R 00 (vier Register je Zeile).

3. Nummer des letzten benutzten Datenspeichers in R 49 eingeben.

4. Programm einlesen. Die Version ab 480 hat den Vorteil, daß Text und Programm auf einer Magnetkarte (Block 3 und 4) gespeichert werden kann.

5. Druckausgabe mit Taste A starten.

6. Ist die gewünschte Zahl von Exemplaren gedruckt, das Programm mit der Taste R/S anhalten.

Bild 2 zeigt ein Druckbeispiel.

Hans Gustav Telgmann, Salvador/Brasilien

```
F U N K S C H A U
   -REDAKTION-

POSTFACH 370120

D 8000 MUENCHEN 37
==================

DEUTSCHLAND
***********
```

Bild 2. Beispiel eines Adressen-Ausdrucks

```
000  76 LBL   017  44 SUM   034  32 X:T
001  11  A    018  48  48   035  43 RCL
002  25 CLR   019  73 RC*   036  48  48
003  69 DP    020  48  48   037  67  EQ
004  00  00   021  69 DP    038  12  B
005  73 RC*   022  03  03   039  01  1
006  48  48   023  01  1    040  44 SUM
007  69 DP    024  44 SUM   041  48  48
008  01  01   025  48  48   042  11  A
009  01  1    026  73 RC*   043  76 LBL
010  44 SUM   027  48  48   044  12  B
011  48  48   028  69 DP    045  98 ADV
012  73 RC*   029  04  04   046  98 ADV
013  48  48   030  69 DP    047  00  0
014  69 DP    031  05  05   048  42 STO
015  02  02   032  43 RCL   049  48  48
016  01  1    033  49  49   050  11  A
```

Bild 1. TI-59-Programm zum Anfertigen von Adressenaufklebern mit dem Drucker PC-100

µC-Nachrichten

Einen 4-bit-Prozessor für *weniger als* **5 US-Dollar** (in Hunderter-Stückzahlen) baut National Semiconductor in zwei Versionen als ROM-lose Mikro-Controller in der N-Kanal-MOS-Technologie und in 40-Pin-Gehäusen. Typische Anwendungen sind Uhren, Schaltuhren, Programmsteuerungen, Kfz-Elektronik, kleine Rechner und elektronische Spiele.

□

Nach Mostek fertigt nun auch Zilog einen **Prototyp-Prozessor,** dessen Gehäuse auf der Oberseite eine EPROM-Fassung enthält. Der Z-8-03 MPE (Mikrocomputer Protopack Emulator), eine ROM-lose Version des Einchip-Mikrocomputers Z-8, dient zur Entwicklung von Programmen zusammen mit einem 2716-EPROM.

□

Nach einer Vereinbarung mit der Western Digital Co. wird **Siemens nun auch Flopy-Disk-Controller** liefern, die zur Typenfamilie WD 176 X gehören. Das Abkommen schließt auch Bausteine ein, die erst noch entwickelt werden. Bei dieser Gelegenheit wurde auch mitgeteilt, daß von den 1,2 Mrd. DM Umsatz des Siemens-Bauelemente-Bereiches rund 500 Mio. DM auf Lieferungen an das eigene Unternehmen entfallen.

□

Texas Instruments stellte kürzlich drei neue ICs für die **Ton- und Geräuscherzeugung** in Mikrocomputer-Systemen vor. SN 76487 N und SN 76488 N sind in IIL-Technik gefertigte Generatoren, die analoge und digitale Schaltungselemente auf einem Chip enthalten. Der Tonerzeugungs-Controller SN 76489 N dient als Interface zu 8- oder 16-bit-Mikrocomputersystemen.

Winfried Fuchs

TI-59 berechnet bewegliche Feiertage

Der programmierbare Taschenrechner TI-59 besitzt in seinem Standard-Software-Modul einen immerwährenden Kalender, der jedoch einen Wunsch nicht erfüllt: die Berechnung der beweglichen Feiertage. Im folgenden wird ein Programm beschrieben, welches dieses leistet. Zuerst wird der verwendete Algorithmus erklärt, so daß es leicht möglich ist, das Programm auch für andere Rechner oder Mikroprozessoren lauffähig umzuschreiben.

Der Algorithmus

Das Konzil von Nizäa legte im Jahre 325 n.Chr. fest, daß das Osterfest am ersten Sonntag nach dem ersten Frühlingsvollmond gefeiert wird. Die anderen beweglichen Feiertage leiten sich vom Osterdatum ab. Alle Angaben stützen sich auf den seit 1583 gültigen Gregorianischen Kalender.

Da eine exakte Berechnung des Osterdatums wegen der komplizierten Mondbewegung sehr schwierig ist, rechnet man heute mit einer vereinfachten Mondbahn. Von C. F. Gauß (1777...1855) wurde eine einfache Regel angegeben, auf der unser Algorithmus basiert.

Das Programm im *Bild 2* enthält auf den Adressen 0...24 ein Unterprogramm zur Berechnung der Modulo-Funktion. Bezeichnet man den Dividend mit a, den Divisor mit b, den ganzzahligen Anteil des Quotienten mit x und den Teilerrest mit c, so ist die Modulo-Funktion $c \equiv \mathrm{mod}\,(a, b)$ definiert. Unser Unterprogramm berechnet den Teilerrest nach der Gleichung $c = b \cdot \mathrm{frac}\,(a/b)$, worin frac (z) die Nachkommastellen der Zahl z bezeichnet.

32363717.	30
3531006400.	31
33212431.	32
2236371731.	33
64000000.	34
23243030.	35
1727211323.	36
3537006400.	37
21353231.	38
2717241523.	39
3113300064.	40
13361523.	41
1735302437.	42
3743321523.	43
1735353235.	44
0.	45
0.	46

Bild 1. Registerinhalte des TI-59

Man berechnet nun zwei Faktoren f und g nach folgenden Gleichungen:
$f = \mathrm{mod}\,\{\lfloor 19 \cdot \mathrm{mod}\,(y, 19)\rfloor + m, 30\}$;
$g = f + \mathrm{mod}\,\{\lfloor 2 \cdot \mathrm{mod}\,(y,4) + 4 \cdot \mathrm{mod}\,(y,7) + 6 \cdot f + n\rfloor, 7\}$.

Hierin bedeuten y die Jahreszahl und für m bzw. n sind die Werte aus *Tabelle 1* einzusetzen.

Der Faktor g muß noch korrigiert werden: ist g = 35, so setze man g = 28. Ist g = 34 und f = 28 und mod (y,19) > 10, so setze man g = 27.

Will man einen der beweglichen Feiertage berechnen, subtrahiert man von g den Wert k aus *Tabelle 2*. Ist das Ergebnis größer als 1, so ist es das gesuchte Datum. Andernfalls addiert man 30.9. Ist das Ergebnis größer als 1, so ist es das gesuchte Datum, andernfalls addiert man noch 29.9.

Eine Ausnahme bildet der Aschermittwoch: man addiert, wenn das Ergebnis kleiner als 1 ist, nicht 30.9, sondern 27.9 + SJ, wobei SJ = 1, wenn das Jahr ein Schaltjahr ist, sonst gilt SJ = 0.

Ein Beispiel:

y = 1979
mod (1979, 19) = 3
m = 24
f = 21
mod (1979, 4) = 3
mod (1979, 7) = 5
n = 5
g = 24

Ostern ist somit am (24 – 8.6) = 15.4.
Chr. Himmelfahrt ist am (24 – 30.4) = (– 6.4), da kleiner als 1: (– 6.4 + 30.9) = 24.5.

Bedienung des Rechners

Wenn ein Drucker PC 100 angeschlossen ist, müssen außer dem Programm auch die Register 30 bis 44 (Block 3) geladen werden.

Geben Sie eine Jahreszahl zwischen 1583 und 2299 ein und drücken Sie Taste A. Der Rechner zeigt das Osterdatum an.

Jetzt können Sie beliebig eine der Tasten B bis E drücken. Diese Tasten haben folgende Funktionen:
B berechnet Aschermittwoch,
C berechnet Christi Himmelfahrt,
D berechnet Pfingsten und
E berechnet Fronleichnam.

Tabelle 1. Die Faktoren m und n

y	m	n
1583–1599	22	2
1600–1699	22	2
1700–1799	23	3
1800–1899	23	4
1900–1999	24	5
2000–2099	24	5
2100–2199	24	6
2200–2299	25	0

Tabelle 2. Subtrahend k

Feiertag	k
Ostern	8.6
Aschermittwoch	23.7
Himmelfahrt	30.4
Pfingsten	20.4
Fronleichnam	9.4

```
000  76 LBL
001  10 E'
002  43 RCL
003  12  12
004  55  ÷
005  43 RCL
006  13  13
007  85  +
008  01  1
009  52 EE
010  94 +/-
011  07  7
012  54  )
013  22 INV
014  59 INT
015  65  ×
016  43 RCL
017  13  13
018  54  )
019  59 INT
020  72 ST*
021  10  10
022  22 INV
023  52 EE
024  92 RTN
025  76 LBL
026  11  A
027  98 ADV
028  99 PRT
029  42 STO
030  09  09
031  42 STO
032  12  12
033  75  -
034  01  1
035  09  9
036  04  4
037  01  1
038  95  =
039  50 I×I
040  32 X:T
041  03  3
042  05  5
043  08  8
044  22 INV
045  77  GE
046  04  04
047  72  72
048  01  1
049  09  9
050  42 STO
051  13  13
052  02  2
053  42 STO
054  10  10
055  10 E'
056  04  4
057  42 STO
058  13  13
059  03  3
060  42 STO
061  10  10
062  10 E'
063  07  7
064  42 STO
065  13  13
066  04  4
067  42 STO
068  10  10
069  10 E'
070  02  2
071  05  5
072  42 STO
073  05  05
074  07  7
075  42 STO
076  06  06
077  43 RCL
078  09  09
079  55  ÷
080  01  1
081  00  0
082  00  0
083  95  =
084  59 INT
085  32 X:T
086  02  2
087  01  1
088  22 INV
089  77  GE
090  01  01
091  25  25
092  67  EQ
093  01  01
094  21  21
095  01  1
096  08  8
097  22 INV
098  77  GE
099  01  01
100  19  19
101  67  EQ
102  01  01
103  15  15
104  01  1
105  07  7
106  67  EQ
107  01  01
108  13  13
109  69 OP
110  35  35
111  69 OP
112  36  36
113  69 OP
114  36  36
115  69 OP
116  36  36
117  69 OP
118  35  35
119  69 OP
120  36  36
121  69 OP
122  36  36
123  69 OP
124  35  35
125  01  1
126  09  9
127  65  ×
128  43 RCL
129  02  02
130  85  +
131  43 RCL
132  05  05
133  95  =
134  42 STO
135  12  12
136  03  3
137  00  0
138  42 STO
139  13  13
140  07  7
141  42 STO
142  10  10
143  10 E'
144  02  2
145  65  ×
146  43 RCL
147  03  03
148  85  +
149  04  4
150  65  ×
151  43 RCL
152  04  04
153  85  +
154  06  6
155  65  ×
156  43 RCL
157  07  07
158  85  +
159  43 RCL
160  06  06
161  95  =
162  42 STO
163  12  12
164  07  7
165  42 STO
166  13  13
167  08  8
168  42 STO
169  10  10
170  10 E'
171  43 RCL
172  07  07
173  85  +
174  43 RCL
175  08  08
176  95  =
177  42 STO
178  11  11
179  32 X:T
180  03  3
181  05  5
182  22 INV
183  67  EQ
184  01  01
185  90  90
186  02  2
187  08  8
188  42 STO
189  11  11
190  03  3
191  04  4
192  22 INV
193  67  EQ
194  02  02
195  17  17
196  02  2
197  08  8
198  32 X:T
199  43 RCL
200  07  07
201  22 INV
202  67  EQ
203  02  02
204  17  17
205  43 RCL
206  02  02
207  32 X:T
208  01  1
209  00  0
210  77  GE
211  02  02
212  17  17
213  02  2
214  07  7
215  42 STO
216  11  11
217  69 OP
218  00  00
219  43 RCL
220  30  30
221  69 OP
222  01  01
223  43 RCL
224  31  31
225  69 OP
226  02  02
227  69 OP
228  05  05
229  01  1
230  32 X:T
231  43 RCL
232  11  11
233  75  -
234  08  8
235  93  .
236  06  6
237  95  =
238  22 INV
239  77  GE
240  02  02
241  44  44
242  99 PRT
243  92 RTN
244  85  +
245  03  3
246  00  0
247  93  .
248  09  9
249  95  =
250  22 INV
251  77  GE
252  02  02
253  56  56
254  99 PRT
255  92 RTN
256  85  +
257  02  2
258  09  9
259  93  .
260  09  9
261  95  =
262  99 PRT
263  92 RTN
264  76 LBL
265  14  D
266  69 OP
267  00  00
268  43 RCL
269  32  32
270  69 OP
271  01  01
272  43 RCL
273  33  33
274  69 OP
275  02  02
276  43 RCL
277  34  34
278  69 OP
279  03  03
280  69 OP
281  05  05
282  01  1
283  32 X:T
284  43 RCL
285  11  11
286  75  -
287  02  2
288  00  0
289  93  .
290  04  4
291  95  =
292  22 INV
293  77  GE
294  02  02
295  44  44
296  99 PRT
297  92 RTN
298  76 LBL
299  13  C
300  69 OP
301  00  00
302  43 RCL
303  35  35
304  69 OP
305  01  01
306  43 RCL
307  36  36
308  69 OP
309  02  02
310  43 RCL
311  37  37
312  69 OP
313  03  03
314  69 OP
315  05  05
316  01  1
317  32 X:T
318  43 RCL
319  11  11
320  75  -
321  03  3
322  00  0
323  93  .
324  04  4
325  95  =
326  22 INV
327  77  GE
328  02  02
329  44  44
330  99 PRT
331  92 RTN
332  76 LBL
333  15  E
334  69 OP
335  00  00
336  43 RCL
337  38  38
338  69 OP
339  01  01
340  43 RCL
341  39  39
342  69 OP
343  02  02
344  43 RCL
345  40  40
346  69 OP
347  03  03
348  69 OP
349  05  05
350  01  1
351  32 X:T
352  43 RCL
353  11  11
354  75  -
355  09  9
356  93  .
357  04  4
358  95  =
359  22 INV
360  77  GE
361  02  02
362  44  44
363  99 PRT
364  92 RTN
365  76 LBL
366  12  B
367  43 RCL
368  41  41
369  69 OP
370  01  01
371  43 RCL
372  42  42
373  69 OP
374  02  02
375  43 RCL
376  43  43
377  69 OP
378  03  03
379  43 RCL
380  34  34
381  69 OP
382  04  04
383  69 OP
384  05  05
385  25 CLR
386  42 STO
387  01  01
388  32 X:T
389  43 RCL
390  03  03
391  22 INV
392  67  EQ
393  04  04
394  19  19
395  43 RCL
396  09  09
397  42 STO
398  12  12
399  01  1
400  00  0
401  00  0
402  42 STO
403  13  13
404  10 E'
405  22 INV
406  67  EQ
407  04  04
408  17  17
409  04  4
410  49 PRD
411  13  13
412  10 E'
413  22 INV
414  67  EQ
415  04  04
416  19  19
417  69 OP
418  21  21
419  01  1
420  32 X:T
421  43 RCL
422  11  11
423  75  -
424  02  2
425  03  3
426  93  .
427  07  7
428  95  =
429  22 INV
430  77  GE
431  04  04
432  35  35
433  99 PRT
434  92 RTN
435  85  +
436  02  2
437  07  7
438  93  .
439  09  9
440  85  +
441  43 RCL
442  01  01
443  95  =
444  99 PRT
445  92 RTN
446  76 LBL
447  16 A'
448  11  A
449  12  B
450  13  C
451  14  D
452  15  E
453  92 RTN
454  76 LBL
455  17 B'
456  11  A
457  69 OP
458  29  29
459  43 RCL
460  09  09
461  61 GTO
462  17 B'
463  76 LBL
464  18 C'
465  16 A'
466  69 OP
467  29  29
468  43 RCL
469  09  09
470  61 GTO
471  18 C'
472  69 OP
473  00  00
474  43 RCL
475  44  44
476  69 OP
477  02  02
478  69 OP
479  05  05
```

Bild 2. Dieses zugegebenermaßen nicht gerade kurze Programm errechnet die beweglichen Feiertage eines beliebigen Jahres zwischen 1583 und 2299

Nach Eingabe einer neuen Jahreszahl müssen Sie stets zuerst mit Taste A durch Berechnung des Osterdatums das Programm initialisieren. Erst dann kann ein anderer Feiertag berechnet werden.

Bei Anschluß eines Druckers PC 100 können Sie nach Eingabe einer Jahreszahl mit Taste A' eine Liste aller fünf beweglichen Feiertage ausdrucken.

Mit Taste B' erhalten Sie eine Liste der Osterfeiertage für die folgenden Jahre. In diesem Fall muß das Drucken mit der Taste R/S gestoppt werden.

Mit Taste C' erhalten Sie eine Liste aller beweglichen Feiertage für die folgenden Jahre. Das Drucken kann mit der Taste R/S beendet werden.

Bei fehlerhafter Eingabe (Jahreszahl kleiner als 1583 oder größer als 2299) blinkt die Anzeige, und das Wort ERROR wird ausgedruckt.

Binär-Dezimal und zurück

Beim Programmieren von externen Geräten über den IEC-Bus oder über einen I/O-Port ist es hilfreich, wenn man Daten nicht in ihrer dezimalen Form, sondern binär eingeben kann. Das Programm in *Bild 1* gestattet dies: Es besorgt die Umwandlung einer Dezimalzahl (die so groß wie der maximale Rechenbereich des verwendeten Computers sein kann) in eine Binärzahl und macht deutlich, wie nützlich dabei die String-Befehle sind. *Bild 2* zeigt ein Programm, das das Gegenteil tut: Eine einzugebende Binärzahl wird in ihr dezimales Äquivalent umgerechnet. Die Binärzahl darf maximal so lang wie der Eingangs-Puffer des Computers (z. B. 80 Zeichen) sein. Fe.

```
LIST

 10 REM DEZIMAL-BINAER-UMWANDLUNG
 20 REM VON HERWIG FEICHTINGER
 30 INPUT"DEZIMALZAHL=";A:H$=" BINAER"
 40 A=A/2
 50 IFA=INT(A)THENH$="0"+H$:GOTO70
 60 H$="1"+H$
 70 A=INT(A):IFA>0GOTO40
 80 PRINTH$:GOTO30
RUN
DEZIMALZAHL=? 255
11111111 BINAER
DEZIMALZAHL=? 1024
10000000000 BINAER
DEZIMALZAHL=? 47654321
10110101110010010110110001 BINAER
DEZIMALZAHL=?
```

Bild 1. Dieses Programm kann dank der Anwendung von String-Befehlen eine nahezu beliebig lange Binärzahl ausgeben

```
LIST

 10 REM BINAER-DEZIMAL-UMWANDLUNG
 20 REM VON HERWIG FEICHTINGER
 30 INPUT"BINAERZAHL=";B$
 40 A=1:C=0
 50 FORI=LEN(B$)TO1STEP-1
 60 IFMID$(B$,I,1)="1"THENC=C+A
 70 A=A+A:NEXT
 80 PRINT"DEZIMAL:";C:GOTO30
RUN
BINAERZAHL=? 10000000000
DEZIMAL: 1024
BINAERZAHL=? 1100010111
DEZIMAL: 791
BINAERZAHL=? 1
DEZIMAL: 1
BINAERZAHL=?
```

Bild 2. Bei der Binär-Dezimal-Umwandlung ist die Länge der Binärzahl nur durch die Länge des Computer-Eingangs-Puffers begrenzt

Kassetten-Interface

Zum Aufzeichnen digitaler Zeichen auf Magnetband sind mehrere Verfahren möglich. Die höchste Störsicherheit dürfte bei Verwendung eines FM-Modems (Modulator-Demodulator) erreicht werden. Der Nachteil hierbei ist nicht nur der relativ hohe Schaltungsaufwand, sondern auch die durch das Modulationsverfahren bedingte kleine Datenaufzeichnungsrate.

Bei dem in *Bild 1* verwendeten Prinzip wird für jedes aufzuzeichnende Bit nur ein Nf-Schwingungszug benötigt, so daß die Anzahl der speicherbaren Bits je Sekunde nur durch die Grenzfrequenz des verwendeten Recorders bestimmt wird.

Zur Aufzeichnung wird für jedes Bit eines Zeichens eine Rechteckschwingung, bestehend aus positiver und negativer Halbwelle, erzeugt. Das Monoflop Q1 wird mit der negativen Flanke eines Bits getriggert und schwingt selbsttätig weiter, solange der Eingangspegel L bleibt. Bei mehreren aufeinander folgenden L-Bits wird also eine fortlaufende NF-Schwingung erzeugt. Um Fehltriggerungen bei kleinen Schwankungen der Kippzeit von Q1 zu vermeiden, erfolgt die Triggerung leicht verzögert gegenüber dem Eingangssignal. Mit den Flipflops Q2 und Q3 wird die Frequenz von Q1 halbiert, wobei Q2 die positiven und Q3 die negativen Halbwellen der Rechteckschwingung am Ausgang erzeugt (*Bild 2*).

Die Zeichenpolarität, bei der Nf-Impulse abgegeben werden, läßt sich durch Umlegen von zwei Brücken im Aufnahme- und Wiedergabeteil umkehren. Im dargestellten Impulsdiagramm wurde als Beispiel das Zeichen für „Space" (hexadezimal 20) im CCITT-Code Nr. 5 gewählt. Es können natürlich auch beliebige andere Codes mit serieller Ausgabe verwendet werden.

Zum Abgleich ist ein Zeichen zu verwenden, in dem sowohl ein einzelnes wie auch eine möglichst lange Reihe direkt aufeinander folgender Bits vorkommen:

Aufnahme:

1. TP 1 auf Masse legen
2. t 2 mit R 1 auf halbe Zeitdauer von t 1 einstellen (Grobabgleich)
3. Masse an TP 1 entfernen
4. t 4 mit R 1 auf gleiche Länge wie t 3 einstellen (Feinabgleich)

Wiedergabe:

1. Mit P 3 Empfindlichkeit so einstellen, daß am Collektor von T 1 saubere Impulse erscheinen
2. Impulslänge t 5 mit R 2 auf gleiche Länge wie t 1 einstellen

Die angegebenen Werte für die Zeitkonstanten der Monoflops gelten für eine Übertragungsgeschwindigkeit von 600 bit/s. Dieser Wert kann natürlich wesentlich erhöht werden. Es sind dann nur die zeitbestimmenden Kondensatoren entsprechend zu verkleinern. Die Schaltung wurde längere Zeit an einer kommerziellen Datenstation getestet. Hierbei traten keine Übertragungsfehler auf. *Horst Lichtenheld*

Tips für TI-59-Benutzer

Viele Programmierkniffe des TI-59 sind nicht bekannt – vor allem, weil sie im Handbuch aus unerfindlichen Gründen verschwiegen werden.

Die Klammerregister 1...8 sind per Programm frei benutzbar. Der dazu notwendige Aufruf-Code ist 82; die Eingabe ist z.B. mit RCL 82 BST BST DEL SST möglich. Der Drucker faßt den Operationscode als „HIR" auf (HIRE: engl. f. mieten; „Mietregister"). Dabei bedeuten:

HIR 01...08	Wert in Klammerregister 1...8 laden
HIR 11...18	Wert aus Klammerregister 1...8 anzeigen
HIR 31...38	SUM in einem der Klammerregister
HIR 41...48	MULT in einem Klammerregister
HIR 51...58	SUB in einem der Klammerregister
HIR 61...68	DIV in einem Klammerregister

Ist die Anzeige nicht im EE- oder ENG-Format und liegt die Zahl zwischen $\pm 0{,}\overline{9}$, so wird bei den letzten vier Operationen die Zahl im Anzeigeregister auf die Form $a \cdot 10^{|b|}$ gebracht, d.h. das Vorzeichen des Exponenten geht dabei verloren.

Der Dsz-Befehl funktioniert entgegen den Angaben im TI-Handbuch für alle Speicher außer 40. Wie man die notwendige Registernummer hinter den Operationscode von Dsz bringt, läßt sich auf tausend Arten lösen – Nachdenken lohnt hier.

Der Befehl „31" bewirkt auch im Programm das Umschalten auf den Programmeingabe-Modus (LRN).

Jens Haase

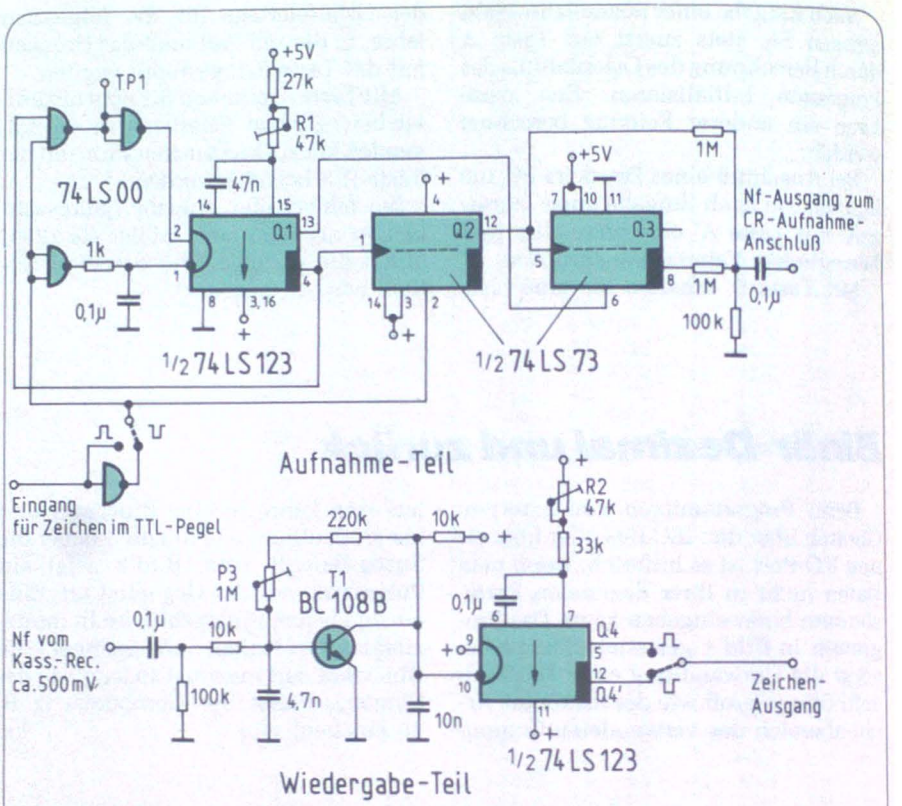


Bild 1. Schaltung des Kassetten-Interface. Die Dimensionierung gilt für 600 bit/s, was auch von einfachsten Kassettenrecordern noch ohne weiteres verkraftet wird

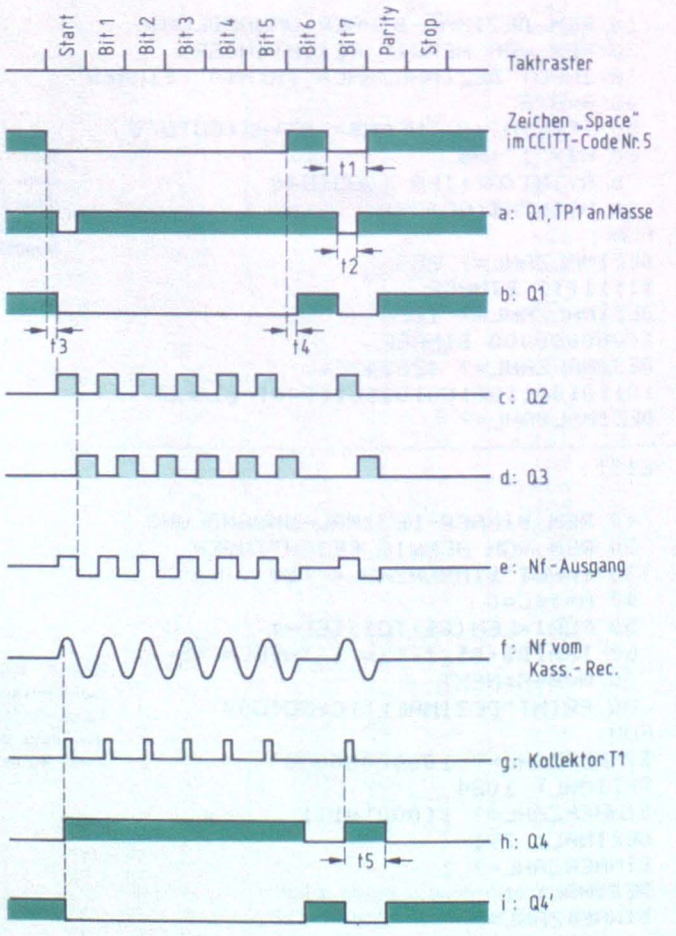


Bild 2. Impulsdiagramm für Aufnahme- und Wiedergabeteil des Modems. Der Kassettenrecorder macht aus den ursprünglich rechteckförmigen Impulsen nahezu einen Sinus

µC-Nachrichten

Die Taufkirchener Firma Schumpich hat die Vertretung für einen 6502-Mikrocomputer übernommen. Er besteht aus zwei Europakarten und ist als Bausatz etwas preiswerter als der KIM-1. Kassettenformat und Bussystem sind allerdings nicht KIM-kompatibel, und auch das Umschreiben von KIM-Programmen auf den „Acorn" ist problematisch, da letzterer keinen Timer enthält. □

Der Mikrocomputer AIM-65 ist von GWK (Herzogenrath) für professionelle Anwender jetzt mit beliebig langer Burn-In-Prüfdauer erhältlich. Damit können auch Langzeitfehler praktisch ausgeschlossen werden. □

Das 16-bit-Mikroprozessor-Kursbuch zum Einplatinen-Computer **TM 990/189** von Texas Instruments liegt jetzt in **deutscher Sprache** vor. Es umfaßt etwa 460 Seiten und beschreibt leichtverständlich die Architektur und Programmierung der Prozessorfamilie 9900. □

HPs Einstieg in den Personal-Computer-Markt mit dem in Heft 3 besprochenen HP-85 ist nicht das letzte Wort des Herstellers. Für Ende des Jahres erwartet man einen **HP-86**, der allerdings auch nicht gerade preiswerter sein soll.

Ralph Tünker

RLC-Netzwerkberechnung mit dem TI-59

Das nachfolgende Programm läuft auf den programmierbaren Taschenrechnern TI-58 und TI-59. Es gestattet die Berechnung des Real- und Imaginärteils der Impedanz eines RLC-Netzwerks.

Bild 1 zeigt das vollständige Programmlisting. Bevor die Werte über die Labels A...D eingegeben werden, muß mit 2nd CMS sichergestellt werden, daß die Speicher 1...15 gelöscht sind. Nach Betätigung des Labels E erscheint der Realteil in der Anzeige. Der Imaginärteil steht im t-Register.

Die richtige Reihenfolge für die Eingabe über die Label bestimmt das zu berechnende Netzwerk. Es können beliebig viele Glieder berechnet werden. Die Berechnungen sind abhängig von der Frequenz; Aus diesem Grund muß sie immer zuerst eingegeben werden. Bei Parallelschaltung eines Gliedes kann die Verarbeitung bis zu 15 Sekunden andauern. Der Programmzeiger befindet sich dann im Standard-Software-Modul des TI-58/59.

Label A Wirkwiderstand in Reihenschaltung [Ω]
Label A' Wirkwiderstand in Parallelschaltung [Ω]
Label B Spule in Reihenschaltung [Hy]
Label B' Spule in Parallelschaltung [Hy]
Label C Kondensator in Reihenschaltung [F]
Label C' Kondensator in Parallelschaltung [F]
Label D Frequenz [Hz]
Label E Ergebnis: Realteil und Imaginärteil [Ω]

Die Datenspeicher 00–05 und 10–15 sind belegt.

```
000  76 LBL   023  94 +/-   046  86 STF   069  42 STO   092  05  05   115  05  05
001  11  A    024  44 SUM   047  01  01   070  13  13   093  15  E    116  11  A
002  44 SUM   025  11  11   048  61 GTO   071  25 CLR   094  42 STO   117  36 PGM
003  10  10   026  25 CLR   049  00  00   072  92 RTN   095  14  14   118  05  05
004  25 CLR   027  92 RTN   050  81  81   073  76 LBL   096  32 X:T   119  15  E
005  92 RTN   028  76 LBL   051  76 LBL   074  15  E    097  42 STO   120  42 STO
006  76 LBL   029  16 A'    052  18 C'    075  43 RCL   098  15  15   121  10  10
007  12  B    030  35 1/X   053  65  ×    076  11  11   099  43 RCL   122  32 X:T
008  65  ×    031  42 STO   054  43 RCL   077  32 X:T   100  12  12   123  42 STO
009  43 RCL   032  12  12   055  13  13   078  43 RCL   101  87 IFF   124  11  11
010  13  13   033  61 GTO   056  95  =    079  10  10   102  01  01   125  25 CLR
011  95  =    034  00  00   057  42 STO   080  92 RTN   103  01  01   126  92 RTN
012  44 SUM   035  81  81   058  12  12   081  43 RCL   104  27  27   127  44 SUM
013  11  11   036  76 LBL   059  61 GTO   082  10  10   105  44 SUM   128  15  15
014  25 CLR   037  17 B'    060  00  00   083  36 PGM   106  14  14   129  22 INV
015  92 RTN   038  65  ×    061  46  46   084  05  05   107  43 RCL   130  86 STF
016  76 LBL   039  43 RCL   062  76 LBL   085  11  A    108  14  14   131  01  01
017  13  C    040  13  13   063  14  D    086  43 RCL   109  36 PGM   132  61 GTO
018  65  ×    041  95  =    064  65  ×    087  11  11   110  05  05   133  01  01
019  43 RCL   042  35 1/X   065  02  2    088  36 PGM   111  11  A    134  07  07
020  13  13   043  94 +/-   066  65  ×    089  05  05   112  43 RCL   135  00  0
021  95  =    044  42 STO   067  89  π    090  11  A    113  15  15
022  35 1/X   045  12  12   068  95  =    091  36 PGM   114  36 PGM
```

Bild 1. Gesamtauflistung des RLC-Programms

Beispiel 1 (Bild 2)

Zuerst wird die Reihenschaltung von R, L und C berechnet. Anschließend der Parallelkondensator, weil dieser der Reihenschaltung parallel liegt.

Eingabe	Label	Anzeige	Bemerkung
CMs			Speicher löschen
80×10^3	D	0	Frequenz
50	A	0	R – Reihe
80×10^6 +/–	B	0	L – Reihe
$12{,}5 \times 10^9$ +/–	C	0	C – Reihe
5×10^9 +/–	C'	0	C – Parallel
	E	29,35953663	Realteil
	X ≷ T	–94,40962144	Imaginärteil

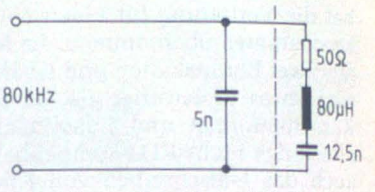


Bild 2. Netzwerk im Rechenbeispiel 1

Beispiel 2 (Bild 3)

Eingabe	Label	Anzeige	Bemerkung
CMs			Speicher löschen
50	D	0	Frequenz
100	A	0	R – Reihe
20×10^6 +/–	C	0	C – Reihe
0,5	B'	0	L – Parallel
	E	246,6338869	Realteil
	X ≷ T	162,1980514	Imaginärteil

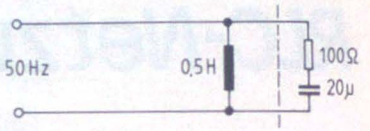


Bild 3. Netzwerk im Rechenbeispiel 2

Beispiel 7 (Bild 4)

Eingabe	Label	Anzeige	Bemerkung
CMs			Speicher löschen
50	D	0	Frequenz
0,5	B	0	L – Reihe
120	A	0	R – Reihe
20×10^6 +/–	C'	0	C – Parallel
50	A	0	R – Reihe
	E	261,0226842	Realteil
	X ≷ T	–155,5054633	Imaginärteil

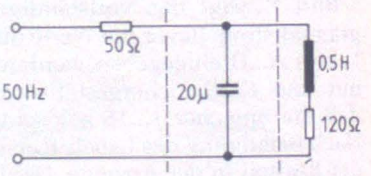


Bild 4. Netzwerk im Rechenbeispiel 3

Primfaktor-Zerlegung mit dem TI-59

Dieses Programm ist sehr nützlich, wenn man in der Digitaltechnik zum Beispiel verschiedene Taktzeiten aus einem Mutteroszillator erzeugen soll, oder einfach aus einer Kiste Quarze den passenden heraussuchen will, aus dem man möglichst einfach herunterteilen kann.

Das Programm ist universell mit und ohne Drucker zu verwenden. Mit Drukker ergibt sich nebenstehender Ausdruck, ohne Drucker werden die Primzahlen nacheinander im Pause-Mode angezeigt. Die letzte Zahl bleibt in der Anzeige stehen.

Das Programm ist sehr benutzerfreundlich aufgebaut, d.h. keine Vorbereitungstasten sind zu drücken. Die zu zerlegende Frequenz wird einfach nach Label A eingegeben, schon läuft alles ab.

Zur Auswertung: Paarweises Vorhandensein von 2en und 5en teilt man mit 10er-Zählern, weitere 2er mit 16er-Zählern, für die übrigen Primzahlen muß man die Zähler entsprechend setzen.

Dipl.-Ing. Fred-H. Heller

```
600600.
     2.
     2.
     2.
     3.
     5.
     5.
     7.
    11.
    13.      ENDE
```

```
000  76 LBL
001  11  A
002  98 ADV
003  99 PRT
004  42 STO
005  01  01
006  02  2
007  42 STO
008  00  00
009  43 RCL
010  01  01
011  76 LBL
012  33 X²
013  55  ÷
014  43 RCL
015  00  00
016  95  =
017  42 STO
018  03  03
019  75  -
020  01  1
021  95  =
022  67  EQ
023  34 Fx
024  22 INV
025  59 INT
026  67  EQ
027  53  (
028  43 RCL
029  01  01
030  42 STO
031  03  03
032  76 LBL
033  54  )
034  69 OP
035  20  20
036  43 RCL
037  03  03
038  61 GTO
039  33 X²
040  76 LBL
041  34 Fx
042  01  1
043  07  7
044  03  3
045  01  1
046  01  1
047  06  6
048  01  1
049  07  7
050  69 OP
051  04  04
052  43 RCL
053  00  00
054  69 OP
055  06  06
056  91 R/S
057  76 LBL
058  53  (
059  43 RCL
060  00  00
061  66 PAU
062  99 PRT
063  43 RCL
064  03  03
065  42 STO
066  01  01
067  69 OP
068  30  30
069  61 GTO
070  54  )
```

TI-58/59-Programm zur Zerlegung einer Zahl in ihre Primfaktoren. Links ist ein Rechenbeispiel zu sehen

Der Nanocomputer: Ein Lehrsystem für Profis

Viele Einkartencomputer werden heute als Lehrsysteme angepriesen. Oft übersehen die Hersteller aber, daß dazu nicht nur geeignete Hardware und Software nötig sind, sondern eine dritte Komponente hinzukommen muß: ausführliche, didaktisch gut aufgebaute Unterlagen. Der von der Firma SGS-Ates angebotene Nanocomputer ist eine der wenigen löblichen Ausnahmen.

Hardware

Den Nanocomputer gibt es in verschiedenen Ausführungen. Grundeinheit ist immer die CPU-Platine mit dem Mikroprozessor Z80 (2,5 MHz), 4 KByte RAM (auf 16 KByte erweiterbar), EPROM/ROM-Fassungen für maximal 8 KByte und zwei parallelen Ein/Ausgabe-Bausteinen (PIOs). Zwei ROMs enthalten normalerweise das Monitorprogramm, und mit einem der PIOs werden Tastatur/Anzeige und serielle Schnittstelle (Kassette, Terminal) betrieben. Zusammen mit einer taschenrechnerähnlichen Ein/Ausgabe-Einheit kostet diese Ausführung 948 DM + MwSt. Etwas mehr, nämlich 1642 DM + MwSt., muß man berappen, wenn man das ganze mit Versuchssteckfeld und Netzteil in einem Gehäuse haben will (*Bild 1*). Ein stolzer Preis, wenn man bedenkt, daß ähnliche Systeme für gleiches Geld schon mit Videointerface zu haben sind.

Hier muß man allerdings das im Preis enthaltene Lehrmaterial berücksichtigen: Auch das kostet Geld, daran wird man sich gewöhnen müssen – besonders, wenn Qualität geboten wird. Zudem ist der mechanische Aufbau (Steckverbindungen, Leiterplatten usw.) durchaus „industriell", und einer späteren Verwendung der CPU-Karte für andere Zwecke steht nichts im Wege.

Die Leiterplatte, auf der das Steckfeld untergebracht ist, enthält drei 40polige IS-Fassungen, an denen sämtliche Bussignale und die PIO-Anschlüsse anliegen. Beim Experimentieren führt man die entsprechenden Signale einfach über Drahtbrücken zum Steckfeld. Als besonders nützlich erweisen sich teildecodierte Adressen, durch die der Adressierungsaufwand bei den Versuchsschaltungen gering bleibt.

Will man den Zustand der Ausgangsleitungen überprüfen oder Signale per Hand eingeben, dann verwendet man hierzu die acht LED-Indikatoren oder die zehn Eingabeschalter, die ebenfalls an einer der IS-Fassungen angeschlossen sind.

Hilfreich für den Lernenden ist die Beschriftung auf der Plexiglasabdekkung. Sie kennzeichnet über der CPU-Karte die verschiedenen Funktionsblöcke und über der Steckfeld-Karte die Signale, die an den IS-Fassungen zugänglich sind.

Bedienung

Grundsätzlich ist der Nanocomputer zum Erlernen der Maschinensprache

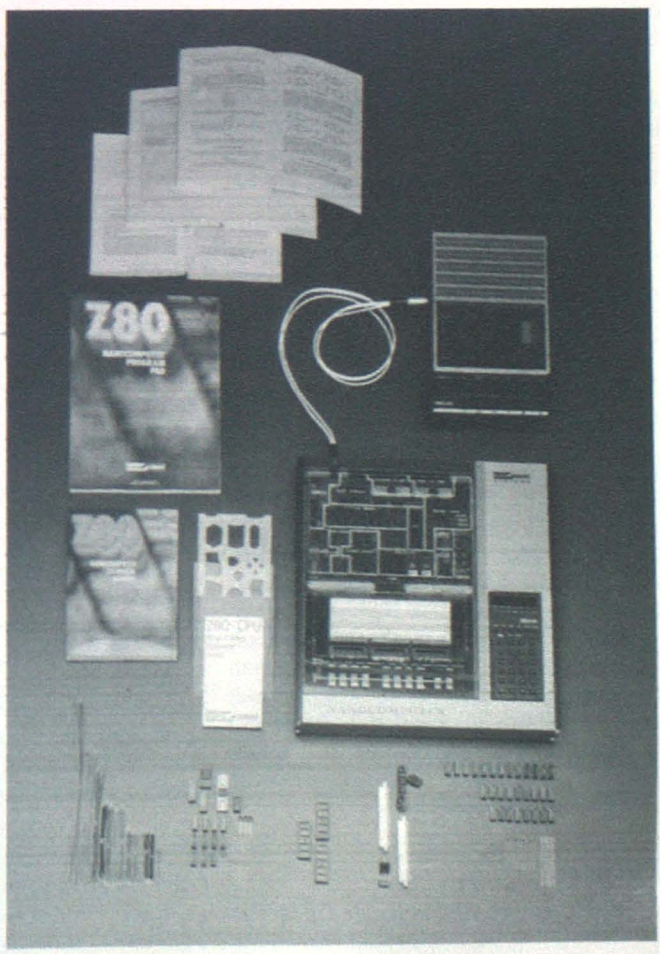


Bild 1. Der Nanocomputer kann ansteckfertig im Gehäuse bezogen werden. Umfangreiche Literatur und ein Drahtsortiment sind bei dieser Version im Preis inbegriffen. Zur Durchführung der Interface-Versuche benötigt der Benutzer noch einige Standard-TTL-Bausteine

geeignet. Als Ein/Ausgabe-Einheit dient ein taschenrechnerähnliches Terminal mit Hexadezimal- und Funktionstasten (*Bild 2*). Neben einem achtstelligen Siebensegment-Display (Daten, Adressen) sind 14 LEDS vorhanden, die über den augenblicklichen Betriebszustand Auskunft geben. Elf von ihnen zeigen an, ob auf dem Display der Inhalt einer Speicherzelle, eines E/A-Ports oder eines internen Registers abgelesen und verändert werden kann. Das Gewünschte wählt der Benutzer aus, indem er mit den Tasten „→" oder „←" diese LEDs der Reihe nach „durchtastet". Die mit ARS bezeichnete Leuchtdiode ist der entsprechenden Taste zugeordnet und zeigt an, ob die Daten im Display aus dem „alternativen Registersatz" (AF' usw.) stammen. Bei einer falschen Eingabe leuchtet die LED „ERR" auf.

Die Funktionstasten spiegeln direkt die Möglichkeiten des Monitors wider: ST, LA, INC und 2ND werden für die üblichen Operationen wie „Speicherzelle anzeigen und verändern", „Register anzeigen und laden" usw. benötigt. DP und LD steuern die Datenübertragung vom und zum Kassettenrecorder (Baudrate programmierbar: 110, 300, 600 Bd). Mit GO startet man ein Programm beim gegenwärtigen Programmzählerstand. BREAK und RESET führen aus jeder Betriebsart in den Monitor zurück. Der Unterschied besteht darin, daß bei RESET alte Registerinhalte zerstört werden, während sie bei BREAK erhalten bleiben.

Es verbleiben die Tasten BRK und SS, die sogenannte Breakpoints und den Einzelschrittbetrieb zulassen. Beides sind wichtige Testhilfsmittel, die in keinem Computer, der für die Programmierung in Maschinensprache vorgesehen ist, mehr fehlen sollten. Für einen seriellen Datenverkehr mit TTY, Drucker, Terminal usw. ist der Monitor ebenfalls ausgelegt. Da ein spezieller UART-Baustein fehlt, sorgt die Software für die Formatumwandlung und benutzt einen der PIOs als Port. Der Nanocomputer kann dadurch dennoch nicht von einem Terminal aus betrieben werden. Will man diese Möglichkeit schaffen, kann man bei der Firma ein anderes Monitorprogramm und einen UART, für den die Verdrahtung bereits vorgesehen ist, bestellen.

Bücher

Das mitgelieferte Lehrmaterial ist der Bestandteil des Systems, der es von anderen unterscheidet und der den relativ hohen Preis rechtfertigt. Es besteht im Fall des größeren Systems aus zwei Büchern in deutscher Sprache: Das eine gibt eine Einführung in die Programmierung auf Maschinenebene, das andere befaßt sich überwiegend mit dem Anschluß externer Schaltungen. Beide gehen nach demselben Rezept vor: Lernen durch Beispiele. Dabei wird vor jedem Versuch das Ziel kurz erklärt. Es folgen – Schritt für Schritt – genaue Angaben, welche Tasten der Benutzer zu betätigen hat und welche Auswirkungen zu erwarten sind. Anhand von Kontrollfragen kann man den erreichten Wissensstand überprüfen.

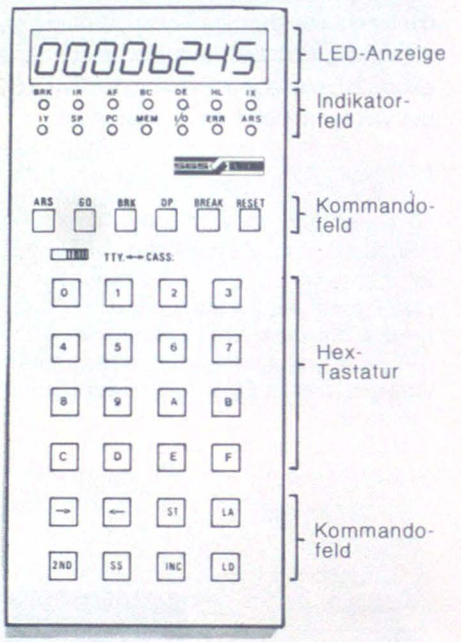


Bild 2. Tastatur- und Anzeigefeld des Nanocomputers

Überspitzt ausgedrückt hält diese Methode, was andere versprechen: Man kann tatsächlich ohne spezielles Vorwissen mit dem Programmieren beginnen. Über eines sollte man sich allerdings nicht hinwegtäuschen: Auch dieser Weg ist beschwerlich – man braucht eine ganze Menge Geduld, wenn man erfolgreich sein will. Die Bücher sind nämlich so aufgebaut, daß man sie nur systematisch von vorne nach hinten durcharbeiten sollte. Einfach mal herumspielen ist ziemlich hoffnungslos.

Man braucht also viel Zeit, wenn man die Vorzüge des Nanocomputers nutzen will. Auch wenn man schon Fachmann ist, muß man Geduld haben, weil bei den Unterlagen eine kurze, übersichtliche Systemdarstellung fehlt.

Dies ist aber der einzige Nachteil im Hinblick auf die Unterlagen. Dem steht eine ganze Reihe angenehmer Seiten gegenüber, z. B. ein komplettes Monitor-Listing mit ausführlichen Kommentaren und die Beschreibung der Unterprogramme, die man auf Wunsch kostenlos erhält.

Die beiden Bücher besitzen noch eine bemerkenswerte Eigenschaft, die nicht selbstverständlich ist. Obwohl sie bei Null beginnen, lassen sie den Leser nicht mit prinzipiellem Wissen zurück, sondern erklären haarklein, wie die Dinge tatsächlich vor sich gehen. Dies gilt vor allem für die wichtigen Interface-Probleme. Der angesprochene Interessentenkreis erweitert sich dadurch um die Entwickler in der Praxis, die eben konkrete Probleme haben und sich nicht mit vereinfachten Darstellungen zufriedengeben können.

Damit für das Eintippen der Software nicht zu viel Zeit verlorengeht, kann man zwei EPROMs mit den Versuchsprogrammen nachrüsten, die man ins RAM übertragen und verändern kann.

Erweiterungsmöglichkeiten

Der Nanocomputer läßt sich auf zwei verschiedene Arten erweitern: zum BASIC-Computer und zum Entwicklungssystem. Im ersten Fall tauscht man das Monitorprogramm gegen den Interpreter aus, schließt ASCII-Tastatur und Videokarte an (beides ist von SGS-ATES erhältlich) und braucht dann nur noch ein Fernsehgerät als Ausgabeeinheit. Das Ergebnis ist mit Maschinen wie PET oder TRS-80 vergleichbar. Auch hierzu gibt es ein Lehrbuch, das ähnlich gut wie die erwähnten Bücher sein soll.

Im zweiten Fall tauscht man den Monitor gegen einen anderen aus, der – wie schon angeschnitten – für TTY-Betrieb ausgelegt ist. Zusätzlich steckt man EPROMs mit dem Editor-Assembler in die Fassungen und setzt den dafür nötigen UART ein. Die CPU-Karte kann jetzt mit einem Terminal betrieben werden. Zusammen mit weiteren Karten (Speicher, E/A-Einheiten, Floppy-Disk-Ansteuerung usw.), die man in einem Baugruppenträger unterbringt, kann man ein System zusammenstellen, das in keiner Weise eingeschränkt ist.

Rudolf Hofer

Johannes Valentini

Ein (fast) ideales Speichermedium

Das Disk-Operating-System (DOS) des Euro-Apple

Nun sind auch auf dem bundesdeutschen Mikrocomputer-Markt die ersten preisgünstigen Floppy-Stationen für Hobbycomputer erschienen; für uns Anlaß, einen Bericht über praktische Erfahrungen mit „Disk II", dem Floppy-Disk-System des „Euro-Apple" ITT 2020, vorzulegen. Besprochen ist die im Spätjahr 1979 erschienene und gegenüber der Vorläufervariante wesentlich verbesserte Version „3.2".

Für eine Vielzahl von Anwendungen von Rechnern im Hobby-Computer-Bereich ist das Speichervolumen, mit dem moderne Steckenpferd-Rechner heute ausgerüstet sind oder auf das sie sich erweitern lassen, bei weitem ausreichend. Aber es bleiben genügend Anwendungsfälle, in denen sich der Programmierer mehr Möglichkeiten zum Ablegen von Programmen oder Daten wünscht. Ein Funkamateur, der seinem Rechner gern die Buchführung über stattgefundene Funkverbindungen übertragen möchte, wird ebenso schnell an die Grenzen seines Speicherzellen-Vorrates stoßen wie beispielsweise ein Hobby-Elektroniker, der seine Literatur-Fundstellen-Kartei von manuellem Betrieb auf eine schnellere und weniger arbeitsintensive Verfahrensweise umstellen möchte.

Zwar bietet sich in solchen Fällen das Deponieren von Daten auf Band an, aber das mühsame Vor- und Rückspulen von Hand beim Einfügen weiterer Informationen in aufgezeichnete Datenblocks sowie die lange Zugriffszeit – sie kann bei umfangreichen Dateien jedesmal viele Minuten umfassen – das ist nicht jedermanns Geschmack. Für datenintensive Anwendungen empfiehlt sich hier als (fast) ideales Speichermedium die Magnetfolienscheibe „Floppy Disk". Floppy-Station und Disk-Operating-System des Euro-Apple ITT 2020 (Version 3.2) werden hier, zusammen mit den Praxiserfahrungen aus einigen Wochen störungsfreien Betriebs, beschrieben.

Die Betriebssysteme und Hochsprachen-Interpreter üblicher Mikrocomputer können von Haus aus mit der Folienscheibe nicht umgehen; zur Sicherstellung korrekten Datenaustauschs zwischen dem Schreib-Lesekopf der Floppystation und dem Rechner bedarf es eines intelligenten Systems, das von sich aus die Buchführung über die niedergelegten Daten vornimmt, genau „weiß", von welcher der konzentrisch auf der Folienscheibe angeordneten Spuren und aus welchem Sektor innerhalb der betreffenden Spur nun Signale abzuholen sind, das darüberhinaus die Fehlerfreiheit der Aufzeichnung überwacht und schließlich noch die Anweisungen des Benutzers zu interpretieren und auszuführen weiß – das Ganze ist Aufgabe des „Disk Operating Systems", des Disketten-Betriebs-Systems. Zur Realisierung nun sind drei unterschiedliche Verfahren im Gebrauch.

Einmal kann man die Aufgabe der Disketten-Verwaltung einem getrennten Mikrocomputer-System in der Diskettenstation selbst übertragen; der Rechner verkehrt mit diesem Subsystem beispielsweise über den IEC-Bus, wie es die Hobbycomputer der Firma Commodore tun. Zweite Möglichkeit: Das Betriebssystem des betreffenden Rechners selbst wird durch Zusatz-ROMs erweitert; einige amerikanische Produkte, die bisher nicht in nennenswerten Stückzahlen auf den deutschen Markt gelangt sind, verfahren so – hierbei tritt eine erhebliche Zusatzbelastung der CPU auf, was sich in der Verarbeitungsgeschwindigkeit bemerkbar macht. Einen dritten Weg gingen Apple Computer Incorporated und Lizenznehmer, die europäischen ITT-Gesellschaften, die den „ITT 2020" vertreiben: Hier ist das intelligente Subsystem zur Laufwerk- und Kopfsteuerung auf einer kleinen Platine angeordnet, die man auf einem der Steckplätze für Erweiterungskarten im Innern des Euro-Apple einfügt. Die Platine enthält Buffer-Bausteine und einen in LSI-Technik ausgeführten „Controller", ein Steuerbauteil, das bis zu zwei Laufwerke bedienen kann. Von der Platine aus führt ein Flachkabel durch die rückwärtigen Kabelschlitze des Rechners zur Floppy-Station. Über die Flachkabel-Adern werden nicht nur Daten und Steuersignale fortgeleitet, sondern es wird auch die Betriebsspannungsversorgung der Station sichergestellt.

Nach Einschalten des Rechners ist der Controller zunächst inaktiv. Erst, wenn durch Tasteneingabe oder im Verlauf der Abarbeitung eines Programms der Befehl PR – gefolgt vom Zeichen # und der Nummer des Steckplatzes mit der Floppy-Steuer-Platine – ergeht, erst dann erwacht der Controller zum Leben und übernimmt nun vorübergehend das Kommando über das Gesamtsystem. Er startet den Antriebsmotor der Diskettenstation – im folgenden wird zur Vereinfachung allein der Fall einer Ein-Laufwerk-Station betrachtet –, positioniert den Schreib-Lese-Kopf über jenen Teil der Magnetfolie, auf der sich vereinbarungsgemäß die Aufzeichnung des Disketten-Betriebssystem-Programms befindet, und lädt sodann die höchsten zehneinhalb Kilobyte des Apple-Speichers mit dem DOS. Danach gibt der Controller die zwangspassivierte CPU des Rechners wieder frei und veranlaßt den Start des Disketten-Betriebsprogramms, das seinerseits das auf der Diskette unter dem Namen „Hello" abgespeicherte Basic-Programm abfährt; hier wird man üblicherweise die Anzeige der Diskettennummer auf dem Bildschirm und weitere Vermerke vorsehen, von denen man wünscht, daß sie beim Erststart gemeldet werden. Jetzt ist der Apple bereit, DOS-Befehle zu befolgen.

Selbst kurzgefaßte Handbücher über Disk-Operating-Systeme umfassen mehrere Dutzend Seiten, es ist an dieser Stelle natürlich nicht möglich, eine erschöpfende Einführung in den Umgang mit dem DOS des Euro-Apple zu geben; stattdessen soll anhand einiger ausgewählter Beispiele ein Überblick über

die Leistungsfähigkeit des Systems gegeben werden.

Als erster aus der Reihe der Beispiel-Befehle sei hier die Anweisung CATALOG genannt. Sie bewirkt den Ausdruck des Inhaltsverzeichnisses einer Diskette, so wie es auf der Scheibe selbst vom DOS fortgeschrieben wird. An einem Beispiel-Ausdruck (*Bild 1*) wird die Struktur dieses Verzeichnisses sichtbar: Auf den Befehl folgt als Überschrift die Ordnungsnummer der Diskette, die man beim Initialisieren der Scheibe festgelegt hat. Sodann sind die Namen der Aufzeichnungen aufgeführt, denen spezielle Vermerke vorangestellt sind, die hier erläutert werden: Ein Sternchen bedeutet, die Aufzeichnung ist „geschützt", das System „weiß", daß es derart geschützte Aufzeichnungen nicht verändern oder gar löschen darf. Es folgt ein Buchstabe, der die Art der Aufzeichnung charakterisiert: I (für „Integer") bezeichnet ein Programm in Ganzzahl-Basic, P (für „Palsoft") eines im Floating-Point-Basicdialekt des „2020", B (für Binary, „Binär") ist das Kennzeichen für Aufzeichnungen in Maschinensprache (hierbei kann es sich um Daten oder Maschinensprache-Programme handeln) und T (für Text) ist das Kennzeichen von Zeichenketten, die Ziffern, Buchstaben und ASCII-Sonderzeichen wie Begrenzer enthalten dürfen.

Nach Ausdruck des Katalogs kann ein beliebiges Programm geladen werden: Durch den Befehl LOAD, gefolgt vom betreffenden Programmnamen.

```
DISK VOLUME 005
*I 002 HELLO
*I 043 PALSOFT
*P 034 WOBBELPLOTTER
*P 003 TEXT AUSLESEN
*P 004 INT
*B 022 INTEGER
 I 054 TALKER-DEMO
 I 016 TALKER
 I 003 TALKER-PREFIX
 B 003 BTALKER
*I 016 DRAGON MAZE
*P 003 HEX NACH DEZ
*P 003 DEZ NACH HEX
 P 002 ALFABET
*P 004 BLOAD-CHECK
 T 002 PROBETEXT
 P 002 X
 P 003 Y
 P 003 WR
 P 002 NEUBILD
 I 053 STARTREK
 P 002 ASSEMBLER-LINKING
 P 002 PASCAL INSTRUCTIONS
 P 002 SPEICHERANALYSE
 P 003 VIETA-UNTERSUCHUNG
 P 004 INTERPRETER-ERWEITERUNG V. 03.1
*B 025 FCL
*B 024 FOCAL
 P 003 COPYCHECK1
 P 003 COPYCHECK2
```

Bild 1. Mit „CATALOG" läßt sich der Inhalt einer Floppy-Disk auflisten

Stellt das DOS fest, daß der zum Programm gehörende Interpreter noch nicht verfügbar ist, so wird er automatisch dazugeladen, sofern der Interpreter selbst auf der Diskette aufgezeichnet wurde: Vor dem Laden eines Palsoft-Programms holt sich der Rechner also selbsttätig erst einmal den Interpreter selbst von der Scheibe. Eine Besonderheit gilt für maschinensprachliche Programme: Holt man sie mit dem Befehl BLOAD von der Diskette, so wird das Programm genau dorthin in die Speicher des Rechners geräumt, wo es sich beim Beschreiben der Diskette befand. Folgt dem Kommando jedoch die Angabe einer Startadresse wahlweise in hexadezimaler oder dezimaler Notation, wird das Programm an den angegebenen Ort verschoben. Außer den schlichten Ladebefehlen LOAD und BLOAD gibt es weiter die Anweisung CHAIN (Kette), die zwar ein Integer-Programm lädt, aber zuvor belegte Variable nicht zerstört: Man kann mit ihnen weiterarbeiten und kommt so zu Programmabläufen von theoretisch unbegrenzter Länge. RUN, gefolgt von einem Programm-Namen, veranlaßt Laden und Starten des Programms.

Freilich findet sich in der neuen Disk-II-Version noch immer eine Unart, die bereits in der Vorläufervariante zu einem bisweilen etwas problematischen Verhalten des Disketten-Betriebssystems führte: Ist das Begrüßungsprogramm „HELLO" in Palsoft geschrieben, so fällt der Rechner 2020 nach der Initialisierung durch eine dergestalt eröffnete Diskette in ein ausgesprochen hysterisches Benehmen: Die Aufgabe, in einem Zug das DOS einzulesen, sich den Palsoft-Interpreter von der Diskette zu holen und schließlich noch das Begrüßungsprogramm „HELLO" abzufahren, überfordert den Rechner. Der Lautsprecher klingelt ohne Unterbrechung Alarmsignale, und der Editor des Palsoft-Interpreters druckt ohne Unterlaß Prompting-Zeichen auf den Schirm. Abhilfe ist glücklicherweise leicht möglich: Wenn man mit der Reset-Taste dem verrücktspielenden Interpreter das Kommando über das System entzieht, meldet sich der Monitor des 2020. Mit dem Betriebssystem-Befehl „3D0G" (Maschinenprogramm beginnend mit Hex-Adresse 030D abfahren) wird die Befehlsgewalt an das DOS und an den Basic-Interpreter zurückgegeben, und fortan gibt's keine Betriebsstörungen mehr. Zurück jedoch zur Vorstellung der DOS-Befehle:

Das Gegenstück zu LOAD und BLOAD sind die Befehle SAVE und BSAVE, gefolgt vom Programmnamen und – im Falle von BSAVE – der Startadresse und der in Byte ausgedrückten Länge des zu ladenden Programms.

Soweit zum Deponieren von Programmen – die Hauptanwendung einer Diskettenstation liegt jedoch in der Aufbewahrung von Daten und Texten. Hierbei gilt folgendes Verfahren: Zur Eröffnung einer Daten-Aufzeichnung durch ein Programm veranlaßt man die Aussendung des nicht druckenden Kontrollzeichens D-Control (LLLL LHLL, ASCII-Zeichen mit dem Dezimal-Binäräquivalent 4); dies ist das Signal für den Rechner, den nachfolgenden Druck-Befehl als Steuerkommando für das DOS zu interpretieren. Mit PRINT „OPEN..." wird sodann die Aufzeichnung eröffnet; die Punkte stehen für den Namen der Datei. Durch die anschließende Anweisung „WRITE", wiederum gefolgt vom Dateinamen, werden nun alle PRINT-Befehle im sich anschließenden Programmsegment solange als Eingabebefehle zum Disk-Beschreiben gewertet, bis die Datei durch CLOSE, gefolgt vom Namen, geschlossen wird. Dies war ein Beispiel zur Anlage einer „ungeordneten Datei", in der eine jede Aufzeichnung von 1 bis zu über 200 Zeichen umfassen durfte, die intern durch ein CARRIAGE RETURN voneinander abgesetzt wurden.

Der Zugriff in diese Datei ist nur sequentiell möglich, eine Eingabe muß nach der anderen gelesen werden. Das DOS des Apple erlaubt jedoch auch wahlfreien Zugriff in Dateien, sofern diese geordnet angelegt wurden: Das heißt, jede Einzelaufzeichnung muß aus der gleichen Zahl von Zeichen bestehen und dem Rechner im OPEN-Befehl die Zeichenzahl pro „Record" als weiterer Parameter mitgeteilt werden. Mehr darüber nach der Besprechung des READ-Befehls, der Entsprechung zu WRITE, die hier folgt:

Ebenso wie WRITE veranlaßt, daß Druckausgaben des Rechners auf die Diskette umgeleitet werden, so sind nach READ – gefolgt wiederum vom Dateinamen – alle GET- und INPUT-Anweisungen nicht länger auf das Tastenfeld bezogen, sondern dienen dem Auslesen von der Diskette. Und nun der wahlfreie Zugriff, die besondere Stärke von DISK II: Folgen der READ-Anweisung als Parameter nach dem Dateinamen die Angabe der Ordnungsziffer der Aufzeichnung in der Datei und dann noch einmal die Ordnungsziffer des Einzelzeichens der betreffenden Aufzeichnung, kann man wahlfrei das Auslesen an jeder beliebigen Stelle beginnen lassen. Dies aber bedeutet nichts anderes, als daß jedes beliebige Byte in

Textstrings innerhalb weniger Millisekunden erreichbar ist – der Grund für die außerordentlich hohe Datenverarbeitungsgeschwindigkeit von DISK II.

Hochinteressante Möglichkeiten eröffnet der DOS-Befehl „EXECUTE". Er bewirkt, daß von der Scheibe geholte Texte vom Betriebssystem so gewertet werden, als liefe der Basic-Interpreter in der Betriebsart „Editing–Immediate", in jenem Betriebszustand also, in dem man Programme eingibt, sie verändert oder indem man unmittelbar Weisungen per Tastenfeld erteilt. Mit Hilfe von EXECUTE-Anweisungen vermag ein übergeordnetes „BIG-BOSS"-Programm selbsttätig beliebige Programme auf der Scheibe zu verändern, sie zu löschen, zu kürzen oder zu neuen Programmblöcken zusammenzusetzen. Faszinierend, daß dabei sogar zwischen verschiedenen Sprachen gewechselt werden kann. Ein übergeordnetes Hauptprogramm entnimmt zum Beispiel, selbst im residenten Dialekt INTEGER-BASIC geschrieben, einem Palsoft-Programm Informationen, um sie umzusetzen und dann einen FOCAL-Interpreter zu modifizieren, der anschließend gestartet wird. Hochkomfortabel weiterhin ein RENUMBER-PROGRAM, das zur Version 3.2 von DISK II mitgeliefert wird: Es erlaubt beim Bearbeiten von Palsoft-Programmen die gezielte Neuordnung der Zeilennummern innerhalb nach Wunsch definierter Blöcke mit wählbarem Anfangswert und frei festzulegendem Zeilennummer-Erhöhungswert. Alle zeilennummer-bezogenen Statements wie GOTO, IF...THEN, GOSUB, ON...GOSUB oder ON...GOTO werden selbsttätig fortgeschrieben, wie bei derartigen Renumierungsprogrammen ohnehin üblich, darüber hinaus – und dies ist neu – bleibt jedoch durchaus die Möglichkeit bestehen, für besonders hervorzuhebende Programmsegmente markante Zeilennummern beizubehalten.

Schließlich hat DISK II auch Programmierern, die auf Maschinensprachen-Ebene arbeiten oder in ihre Hochsprachen-Programme Maschinenroutinen einzufügen gedenken, einiges zu bieten: DISK II erlaubt, auf dem Weg über die Benutzung einer bequem zugänglichen Subroutine „RWTS" (Read or Write a Track or Sector, unmittelbares Schreiben auf oder Lesen von einzelnen Spuren oder Sektoren der Diskette) jede Magnetscheibe beliebig zu manipulieren – etwa dann, wenn es gilt, einem selbstgeschneiderten Spezialsprachen-Interpreter mit der Fähigkeit zu versehen, sich seinerseits der Diskette zu bedienen. Lob verdient schließlich noch das (englische) Handbuch zu DOS II: Ein vorzüglich gemachter „Floppy-Lehrgang".

An einem konkreten Programmbeispiel (*Bild 2*) wird nun erläutert, wie praktisch realisiert wird, was oben generell dargestellt wurde. Das Palsoft-Programm DISKTEXT AUSLESEN erlaubt, zu Prüf- und Experimentierzwecken den Inhalt eines jeden beliebigen Text-Records von der Diskette zu holen und auf dem Schirm sichtbar zu machen. In Zeile 30 wird unter dem Namen D-String das nichtdruckende Sonderzeichen zur DOS-Steuerung bereitgestellt, sodann der Name der abzufragenden Datei eingeholt und die Anzahl der darzustellenden Aufzeichnungen (Blocks, Records) erfragt. Die DOS-Kommandos in Zeile 60, in der obigen Übersicht nicht näher besprochen, dienen – wie ihre Entsprechungen in Zeile 230 – der Einschaltung einer speziellen Betriebsart und ihrer späteren Rücknahme: Input, Output und Commands sollen nicht länger über den Schirm laufen. In Zeile 80 wird auf die Datei mit Namen in F-String als Datenquelle umgeschaltet, die in Zeile 100 beginnenden Schleife holt die Zeichen von der Scheibe und druckt sie in Zeile 110 aus. In Zeile 240 wird die eröffnete Datei wieder geschlossen.

Außer den beim Einsatz von Palsoft-Begrüßungsprogrammen beobachteten Unzuverlässigkeiten zeigte DISK II einen weiteren Schönheitsfehler: Enthält eine Zeichenkette gewisse Sonderzeichen, die offenbar als interne Begrenzer dienen, gelingt das Zurücklesen nur mit Hilfe eines Tricks: Beim Schreiben des Strings auf Kassette ist die Kette durch das ASCII-Dezimaläquivalent von Anführungszeichen zu eröffnen und zu begrenzen. Geschieht dies nicht, so kommen nur Bruchstücke zurück und das System meldet „Extra ignored" – so, als habe man bei manueller Eingabe im Verlauf einer INPUT-Operation versehentlich mehr Daten angeboten, als das Programm erwartet: Ein Mangel, der nur beim Auslesen per INPUT erfolgte, nicht jedoch bei der Datenbeschaffung durch GET störend in Erscheinung trat.

Abgesehen von beiden Mängeln, die eher in der Klasse der Schönheitsfehler denn unter schwerwiegenden Systemfehlern einzuordnen sind, arbeiteten Floppy-Station und DOS des Euro-Apple im Verlauf von mehr als 200 Betriebsstunden mit absoluter Zuverlässigkeit, und zwar auch bei der Lösung von Aufgaben mit professionellen Anforderungen wie bei der Bearbeitung von Material eines Statistischen Landesamtes zur Anfertigung von Trendanalysen.

```
JLIST
10  REM DISKTEXT AUSLESEN
20  REM ----------------
30 D$ =  CHR$ (4)
40  INPUT "FILE-NAMEN? ";F$
50  INPUT "ZAHL DER BLOCKS? ";Z
60  PRINT D$"NOMONI,O,C"
80  PRINT D$"READ"F$
100  FOR N = 1 TO Z
110  INPUT A$: PRINT A$
120  NEXT
200  PRINT D$
210  PRINT : PRINT "FORTSETZUNG?
     (J/N)"
220  GET A$: IF A$ = "J" THEN 50
230  PRINT D$"MONI,O,C"
240  PRINT D$"CLOSE"
250  PRINT " +++ BEENDET. "
```

Bild 2. Einfaches Programm zum Auslesen der auf einer Floppy gespeicherten Texte

FUNKSCHAU-Sonderheft:

Programme für Kleincomputer und Taschenrechner

Ein Mitte April erscheinendes FUNKSCHAU-Sonderheft soll Besitzern von programmierbaren Taschenrechnern, Basic-Maschinen und Einplatinen-Computern ein wenig aus der Software-Patsche helfen. Hier einige interessante Beiträge des neuen FUNKSCHAU-Sonderheftes:

Flugvorbereitung – ein Basic-Programm für das Errechnen von Kurs, Spritmenge und anderer Daten bei Überlandflügen; AIM-65-Plotter – ein Beweis, daß ein Thermodrucker mehr kann als nur Buchstaben und Zahlen auszugeben; Lohn- und Einkommensteuerberechnung mit dem HP-67; 6502-Disassembler; 6502-Texteditor; Morsezeichen-Decoder in Basic und in 6502-Maschinensprache; Funkfernschreibprogramm mit AIM-65-Druckerausgabe; Spulenberechnung mit dem TI-59; HP-41 druckt Kalender; TRS-80 macht Musik; Zahlenumwandlung mit dem TI-59; Software-Timing ersetzt Bandzählwerk; Hex-BCD-Umwandlung mit dem Z-80; Basic-Computer als Notenassistent für Lehrer und Schüler; Berechnung von Yagi-Richtantennen in Basic... und vieles andere mehr.

Das Sonderheft wird ab Mitte April in allen guten Buch- und Zeitschriftenhandlungen, in Elektronik-Fachgeschäften oder auch direkt beim Franzis-Verlag für 14.80 DM (plus 1.50 DM Porto) erhältlich sein.

Die „nullte Seite" auf einen Blick

Das ursprünglich auf dem 6502-Mikrocomputer AIM-65 entwickelte Programm in *Bild 1* erstellt eine nützliche Referenzliste, die angibt, wo in einem Programm welche Zero-Page-Zellen benutzt werden. Dies ist besonders nützlich, wenn man vorhandene, u.U. schlecht dokumentierte Software ändern möchte. Dabei ist lediglich noch zu beachten, daß natürlich nur jene Zero-Page-Zellen als benutzt erkannt werden, die wirklich einem Z-Page-Operationscode folgen. Deswegen wertet das Programm z.B. nicht die im Befehl LDA 0023 benutzte Zelle 23 aus, und ebenso nicht jene Zellen, die indiziert mit einem Adressenversatz ungleich Null angesprochen werden. Trotz dieser Einschränkung kann man sich schnell einen Überblick über die Belegung der Zero-Page verschaffen; *Bild 2* zeigt, wie das Programm von Bild 1 eine Z-Page-Referenzliste für sich selbst erstellt.

```
0000 A9 00 85 EC
0004 A2 0A BD BA
0008 00 20 C7 00
000C CA 30 11 E0
0010 03 D0 F3 20
0014 CD 00 85 EB
0018 20 CD 00 85
001C EA 4C 06 00
0020 20 CD 00 85
0024 EE 20 CD 00
0028 85 ED A5 EA
002C 85 EF A5 EB
0030 85 F0 A0 00
0034 A2 07 B1 EF
0038 3D 9F 00 5D
003C A6 00 F0 03
0040 CA D0 F3 BD
0044 AE 00 10 37
0048 A2 03 B1 EF
004C 29 1F 5D B6
0050 00 F0 2A CA
0054 10 F4 A0 01
0058 B1 EF C5 EC
005C D0 1F 20 D0
0060 00 A5 F0 20
0064 CA 00 A5 EF
0068 20 CA 00 20
006C C5 00 A0 00
0070 B1 EF 20 CA
0074 00 20 C5 00
0078 A5 EC 20 CA
007C 00 A9 02 AA
0080 E6 EF D0 02
0084 E6 F0 A5 EF
0088 C5 ED D0 06
008C A5 F0 C5 EE
0090 F0 05 CA D0
0094 EB F0 9B E6
0098 EC F0 02 D0
009C 8D 4C 82 E1
00A0 0C 1F 0D 87
00A4 1F FF 03 0C
00A8 19 08 00 10
00AC 20 03 FF 03
00B0 03 01 01 FF
00B4 03 01 00 02
00B8 09 10 3D 4F
00BC 54 20 3D 4D
00C0 4F 52 46 00
00C4 0D A9 20 4C
00C8 BC E9 4C 46
00CC EA 4C FD E3
00D0 4C F0 E9
```

Bild 1. Dieses 6502-Programm erstellt eine Zero-Page-Referenzliste. Die Startadresse ist 0000

Eine Adaption an andere 6502-Systeme ist hier besonders leicht möglich, da alle Monitor-Unterprogramme indirekt über folgende Adressen angesprungen werden:

00C7 Ausdrucken eines ASCII-Zeichens
00CA Ausdrucken eines Bytes (2 Zeichen)
00CD Eingabe eines Bytes (2 Zeichen)
00D0 Ausgabe von CR/LF
009D Rücksprung zum Monitor

Beim KIM-1 ist daher ab 00C7 zu schreiben: 4C A0 1E 4C 3B 1E 4C 9D 1F 4C 2F 1E. Bei 009D ist 4C 4F 1C einzusetzen. Die Startadresse des Programms ist 0000; der Ausdruck der Referenzliste beinhaltet die Adresse des Operationscodes, der auf die Zero-Page zugreift, den Operationscode selbst und die Z-Page-Adresse. Bei FROM und TO ist – stets vierstellig – der Bereich des zu untersuchenden Programms einzugeben, wobei das Durchsuchen von Tabellen vermieden werden sollte, weil es zu mißverständlichen Ergebnissen führen kann. *Herwig Feichtinger*

```
<G>/
FROM=0000 TO=00A1
001B 85 EA
002A A5 EA
0016 85 EB
002E A5 EB
0002 85 EC
005A C5 EC
0078 A5 EC
0097 E6 EC
0028 85 ED
0088 C5 ED
0023 85 EE
008E C5 EE
002C 85 EF
0036 B1 EF
004A B1 EF
0058 B1 EF
0066 A5 EF
0070 B1 EF
0080 E6 EF
0086 A5 EF
0030 85 F0
0061 A5 F0
0084 E6 F0
008C A5 F0
<
```

Bild 2. Hier hat das Programm in Bild 1 von sich selbst eine Referenzliste ausgedruckt; von links nach rechts: Referenzadresse, Operationscode, Zero-Page-Adresse. Die Liste ist nicht nach Programmadressen, sondern nach Zero-Page-Zellen geordnet

Hexadezimal-dezimal und zurück – ganz einfach

Die Mikrocomputer AIM-65 und PC-100 gestatten, wenn sie mit Basic-ROMs ausgerüstet sind, die Umwandlung hexadezimaler in dezimale Zahlen und umgekehrt auf höchst einfache Weise. Dabei wird die Tatsache ausgenutzt, daß die Zeilennummern eines Basic-Programmes hexadezimal in Form von zwei Bytes an den Adressen 0214 (L) und 0215 (H) gespeichert werden. Man initialisiert also – wie gewöhnlich – das Basic und schreibt irgendein Mini-Programm, bestehend aus einem Befehl, mit einer dezimalen Zeilennummer (1...65535), zum Beispiel 489 PRINT (Return).

Verläßt man mit der Escape-Taste jetzt Basic, so kann man durch Drücken von M 214 (Return) die hexadezimale Zeilennummer sehen: 0214 E9 01. Dezimal 489 ist also hexadezimal 01E9. Umgekehrt kann man jetzt an diese beiden Adressen eine andere Hexzahl schreiben – nicht vergessen: das niederwertige Byte zuerst – und mit Taste 6 in das Basic zurückgehen. Der LIST-Befehl macht jetzt die dezimale Zeilennummer sichtbar, die der eingegebenen Hex-Zahl entspricht.

Ein im Speicher stehendes Basic-Programm wird nicht zerstört; man sollte lediglich nach solchen Spielereien an die Adressen 214 und 215 wieder das schreiben, was ursprünglich darin stand, und nie versuchen, mit Taste 5 in das Basic zurückzugelangen, da sonst das Basic-Programm verloren ist. Übrigens wäre es verwunderlich, wenn das bei anderen Computern nicht funktionieren würde; bei einem KIM-System mit Tiny Basic wurde es mit Erfolg bereits auch ausprobiert. Fe.

Ing. (grad.) Gerhard Bartels

MZ-80 K – ein Computer aus Japan

Nach vielen anderen hat jetzt auch Sharp einen Personal-Computer auf den Markt gebracht: den MZ-80K. Er ist eine komplette Einheit, bestehend aus Tastatur, Monitor und Kassettenrecorder. CPU ist die bewährte Z-80.

Standardmäßig ist der MZ-80K mit 20 KByte-RAM ausgestattet und bis 48 KByte ausbaufähig. Als Programmiersprachen stehen Basic und Maschinensprache „beide auf je einer Kassette" zur Verfügung.

Aufbau des Systems

Das Gerät macht optisch einen soliden Eindruck (*Bild*). Als Standfläche benötigt es (B x T) 410 x 470 mm. Die Integration von Monitor und Kassetten-Recorder ist optisch und mechanisch gelungen. Nach dem Lösen von 4 Schrauben läßt sich das gesamte Gehäuse-Oberteil hochklappen und gewährt einen Einblick in den sauberen Aufbau. Sämtliche Einheiten sind leicht zugänglich und über Flachkabel mit der Hauptplatine verbunden.

Im Unterteil befindet sich im rechten Drittel das gekapselte Schalt-Netzteil, während der restliche Platz von der Hauptplatine ausgefüllt wird. Hier sind vorne gut zugänglich die RAM-Sockel angeordnet. Alle wichtigen ICs befinden sich in Steckfassungen.

Die Tastatur

Die Tastatur besteht aus 78 Einzeltasten, die in drei farblich unterschiedliche Bereiche eingeteilt sind: grau für Buchstaben (groß/klein), Ziffern, Zeichen und Grafik-Symbole; blau für Grafik-Symbole; gelb für Editier-Funktionen, die mit dem Cursor ausgeführt werden.

Der größte Teil der Tasten verfügt über eine 3fach-Belegung, so daß 211 verschiedene Zeichen eingegeben werden können (über den POKE-Befehl sogar 252). Die Break-Taste ist über die Shift-Taste verriegelt. Die Beschriftung ist gut leserlich und durch Plastikkappen geschützt. Die Tasten sind groß und leichtgängig. Die Entprellung ist softwaremäßig sehr gut durchgeführt; mit dem POKE-Befehl läßt sich eine akustische Eingabekontrolle einschalten.

Durch den leicht abgeschrägten Einbau ist ein ermüdungsfreies Arbeiten über längere Zeit möglich. Ein Nachteil ist das Fehlen einer Reset-Taste, obwohl die Anschlüsse dafür auf der Hauptplatine vorhanden sind – das hat der MZ-80K mit dem PET-2001 gemeinsam.

Der Monitor

Der Bildschirm hat eine Diagonale von 10 Zoll und ist durch eine Plastikscheibe geschützt. Es lassen sich 25 Zeilen mit je 40 Zeichen darstellen. Die Zeichen werden im 8 x 8 Raster abgebildet. Die 1000 Abbildungsplätze lassen sich auch gezielt mit dem POKE-Befehl ansprechen. Zusätzlich kann man über die Befehle SET und RESET eine 4000-Punkte-Grafik erzeugen.

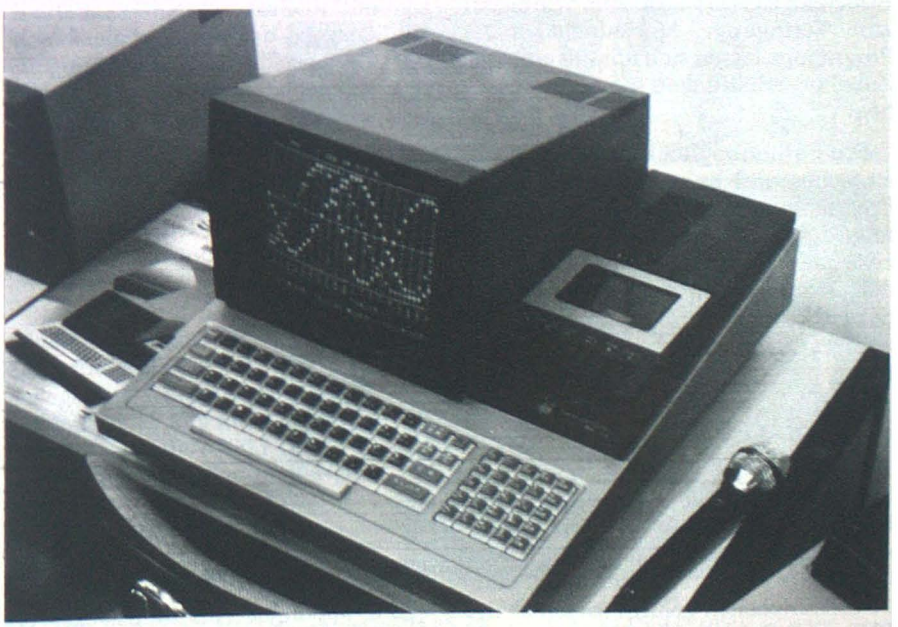
Nach dem Hochklappen des Gehäuse-Oberteils kommt man bequem an die 6 Einstellregler für Lautstärke, Kontrast, Helligkeit, Fokus, Vertikal- und Horizontalsynchronisation heran.

Der Kassettenrecorder

Der eingebaute Kassettenrecorder ist in seiner Aufnahme- und Wiedergabetechnik speziell für die Übertragung von Daten ausgelegt. Ein eingebautes Bandzählwerk gestattet die bequeme Benutzung von C-60-Kassetten. Eine lobenswerte Einrichtung ist der softwaremäßige Stopp des Bandes nach dem Abspeichern oder Einlesen; ähnlich wie beim PET. Der Recorder besitzt auch die für den normalen Gebrauch nötigen Bandlauftasten. Die Übertragungsgeschwindigkeit beträgt 1200 Baud.

Die Hauptplatine

Die Hauptplatine im unteren Teil des Gehäuses ist der eigentliche „Computer". Hier findet man als „Gehirn" die CPU Z-80, sowie die Speicher-RAMs, Video-RAM, Zeichen-ROM, MONITOR-ROM, Ein-Ausgabesteuerung, Zeitgeber und Taktgenerator, alles übersichtlich und sauber aufgebaut.



Der MZ-80 stellt offensichtlich die japanische Antwort auf PET und TRS-80 dar

Tabelle 1: Speicheraufteilung beim MZ-80 K

0000-0FFF	Monitorprogramm-ROM
1000-11FF	Monitor-RAM (reserviert)
1200-5FFF	RAM (Grundversion)
6000-CFFF	RAM-Erweiterung
D000-DFFF	Video-RAM-Bereich
E000-FFFF	Reservierter Bereich

Die CPU Z-80 wird mit einem 2-MHz-Takt betrieben, also nicht gerade besonders schnell! Sämtliche CPU-Anschlüsse sind erst gebuffert, bevor sie weitergeleitet werden. Der Adreßbereich erstreckt sich über 64 KByte und gliedert sich nach *Tabelle 1* auf.

Der Anwender-Speicherbereich ist minimal auf 20 KByte ausgebaut und kann auf maximal 48 KByte erweitert werden (abzüglich des BASIC-Platzes).

Der Zeichengenerator enthält 2 KByte, das Video-RAM 1 KByte. Außerdem befindet sich auf der Platine ein programmierbarer Tongenerator. Das Monitor-Programm hat einen Umfang von 4 KByte.

Die Software

Fest eingebaut in einem ROM steht dem Anwender ein 4 KByte Monitorprogramm zur Verfügung, das die Editier-Befehle und die Cursor-Steuerung beinhaltet. Außerdem hat man nur noch die Befehle LOAD zum Laden von Basic, GOTO$ 1200 zum Neustart von Basic und GOTO$ 1260 zum Warmstart zur Verfügung. Maschinensprache-Programme lassen sich nur mit einer als Zubehör erhältlichen Kassette eingeben!

Die Editiermöglichkeiten lassen ein einfaches und bequemes Ändern von Programmen zu. Mit der CR-Taste erfolgt die Eingabe in den Rechner, und zwar jeweils eine ganze Zeile, wo sich der CURSOR auch immer befindet. Ist der Bildschirm vollgeschrieben, so wird er automatisch nach oben „gerollt".

Das Sharp-Basic

Das Basic von Sharp ist eine Eigenentwicklung. Es konnten bisher keine Fehler des BASIC-Interpreters festgestellt werden. So kennt er auch nicht die in FUNKSCHAU 1980, Heft 2, auf Seite 79 beschriebenen „Null-Probleme" des PET und Euro-Apple (mit Microsoft-Basic). Abweichungen in der 8. Stelle um ± 1 Bei Zahlen ab 512 werden, z. B. bei IF-Abfragen, gerundet.

Das Basic wird nur auf Kassette geliefert, was den Nachteil hat, daß es nach jedem Rechner-Einschalten neu geladen werden muß (Dauer: 1 min. 45 s), zum anderen hat man aber den Bereich für Maschinensprache-Programme zur Verfügung, wenn man nur die Masch.-Spr.-Kassette lädt.

Das Basic ist erstaunlich lang, nämlich fast 14 KByte! Trotzdem steht es in seiner Leistungsfähigkeit nur zwischen PET und TRS-80-Level 2. *Tabelle 2* zeigt den kompletten Befehlssatz. In der Grundausbaustufe stehen dem Anwender immerhin noch 6 KByte zur Verfügung. Eine Besonderheit stellt der MUSIC-Befehl dar. Mit ihm kann man Musik nach Noten programmieren. Als Variablen-Namen sind max. zwei alphanumerische Zeichen zulässig, wovon das erste ein Buchstabe sein muß. String-Variable dürfen max. 255 Zeichen lang sein. Felder sind zweidimensional möglich (max. 256 x 256). File-Namen dürfen max. 16 Zeichen umfassen. Der Zahlen-Bereich erstreckt sich von etwa 0.9E+ 19 bis 0.2E–19. Bei physikalischen Berechnungen ist also etwas Vorsicht angebracht. Die Ergebnisse werden 8stellig angezeigt und verarbeitet.

Neben Programmen können auch Daten mit oder ohne File-Namen abgespeichert und eingelesen werden. Der Vorgang ist sehr einfach zu handhaben. Das Abspeichern geschieht mit WOPEN,,,,..."-PRINT/T...-CLOSE, das Einlesen mit ROPEN,,,,..."-INPUT...-CLOSE. Die Ausgabe der Daten erfolgt blockweise, wobei die Block-Steuerung intern vorgenommen wird. Leider ist diese allerdings sehr sichere Art sehr zeitraubend: WOPEN/ROPEN 14 sec, Block 7,4 sec (5,6 sec Synchr. + 1,8 sec Daten), CLOSE 7 sec, wobei ein Block ca. 256 Bytes lang ist. Übertragungsfehler sind selbst mit Billig-Kassetten nicht aufgetreten.

Eine interne Uhr kann mit TI$ gesetzt und abgefragt werden und mit TM$ kann eine Vergleichszeit vorgegeben werden. Außerdem gibt es die Befehle PRINT/P und LIST/P für die Ausgabe an einen extern anschließbaren Drukker.

Was noch auffiel

Ein 50poliger Anschlußstecker ermöglicht den Anschluß eines Drukkers oder einer inzwischen erhältlichen Interface-Box mit 8-bit-parallel-Port und V-24-Schnittstelle. Da das Basic aber keine IN- und OUT-Befehle kennt, können Ein- und Ausgaben nur in Maschinensprache erfolgen (beim PET kann dies mit PEEK und POKE geschehen, da die I/O-Ports bei ihm „Memory mapped" sind.)

Außer einem Basic-Handbuch in lustiger Aufmachung, aber in deutsch und gut geschrieben, liegt dem Gerät keine weitere Literatur bei: Man kann ja doch nur in Basic programmieren, sagt Sharp! Es ist auch nicht so ohne weiteres möglich, Einsicht in das Basic zu nehmen, es ist geschützt.

Der allgemeine Betrieb mit dem Rechner hat bisher keine Mängel erkennen lassen. Die Wärmeentwicklung bei Dauerbetrieb ist gering. Der einzige Mangel ist, daß ein so großes BASIC nur so wenig kann! Wünschenswert wäre ein „Extended Basic", das auch Befehle wie IN, OUT, RENUMBER und APPEND kennen würde.

Tabelle 2: Basic-Befehlssatz

Befehle
NEW-CLR-RUN-CONT-BYE-SIZE-LIST-LIST/P

Anweisungen
TAB-SPC-DIM-STOP-END-REM-USR-LIMIT-RESTORE-RETURN-DEF FN-READ-DATA-GOTO-GOSUB-ON... GOTO-ON...GOSUB-IF...THEN...-IF... GOTO...IF...GOSUB-FOR... STEP...NEXT-MUSIC-TEMPO-TI$-TM$

Ein-Ausgabe-Befehle
PRINT-PRINT/P-INPUT-GET-PEEK-POKE-SET-RESET

Operations-Symbole
= + –* / ↑

Vergleichsoperatoren
= >= <= <> >< +(ODER) * (UND)

Arithmetische Operatoren
ABS-SGN-INT-SIN-COS-TAN-ATN-SQR-EXP-LOG-LN-RND

String-Funktionen
LEFT$-RIGHT$-MID$-LEN-ASC-CHR$-VAL-STR$

Kassetten-Funktionen
LOAD-SAVE-VERIFY-WOPEN-PRINT/T-ROPEN-INPUT/T-CLOSE

weitere Symbole
? (≷ PRINT) : ; , ... ,,,,..." $ π (=63,14...).

Reinhard Wilckens

Video-Anschluß für den AIM-65

Ein Fernsehmonitor als zusätzliches Ausgabegerät für den AIM-65 bietet vor allem beim Arbeiten mit Basic viele Vorteile. Auch beim Programmieren in Maschinensprache lernt man die Vorzüge bald schätzen; insbesondere läßt sich im Step-Modus der Ablauf von Programmen papiersparend verfolgen.

Das im folgenden beschriebene Systemkonzept ergab sich aus folgenden Forderungen:

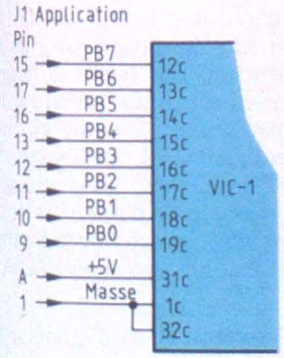
1. Verwendung einer handelsüblichen Video-Interface-Karte
2. Bereitstellung eines universell verwendbaren Unterprogramms zur Zeichenausgabe und Cursor-Positionierung
3. Darstellung sämtlicher AIM-Monitor-, Assembler- und Basic-Funktionen einschließlich der Delete-Funktion, auch im STEP-Betrieb auf dem Bildschirm
4. Keine Beeinflussung maskierbarer Interrupts
5. Wahlweises Zuschalten des TV parallel zu Display und Printer
6. Keine Benutzung von User-RAM für Zwischenspeicher, d.h. Programm vollständig in EPROMs lauffähig
7. Speicherbedarf ≦ 256 Byte

Hardware

Zur Ansteuerung des Fernsehgerätes wurde die Video-Interface-Karte VIC-1 der Firma Eltec (Neubrunnenstr. 10, 6500 Mainz) verwendet, die betriebsfertig für weniger als 230 DM erhältlich ist. Damit ist die Darstellung von 16 Zeilen à 64 Zeichen (Großbuchstaben, Ziffern und Sonderzeichen) möglich. Sie wird gemäß *Bild 1* an den User-Port B angeschlossen.

Eine zusätzliche kleine Schaltung ermöglicht den Betrieb im STEP-Mode und die Benutzung der Delete-Taste (*Bild 2*). Sie puffert gleichzeitig die Adressen für eine eventuelle Systemerweiterung.

Der Schaltungsteil A dekodiert die Adressen xF00 bis xFFF und sperrt in diesem Speicherbereich die Erzeugung von NMI-Signalen für den STEP-Mode. Bei der 4-K-Version des AIM wird die TV-Software zweckmäßigerweise ab 0F00 geladen. Bei der Relokation des



◄ Bild 1. Anschluß der Interface-Karte an den Mikrocomputer AIM-65. Selbstverständlich arbeitet die Schaltung auch mit dem PC-100 von Siemens

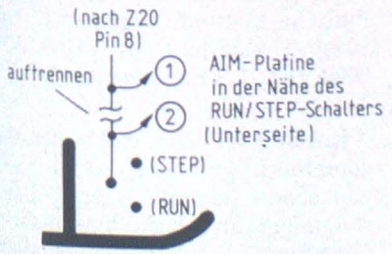


Bild 3. Anschluß des Schaltungsteils A

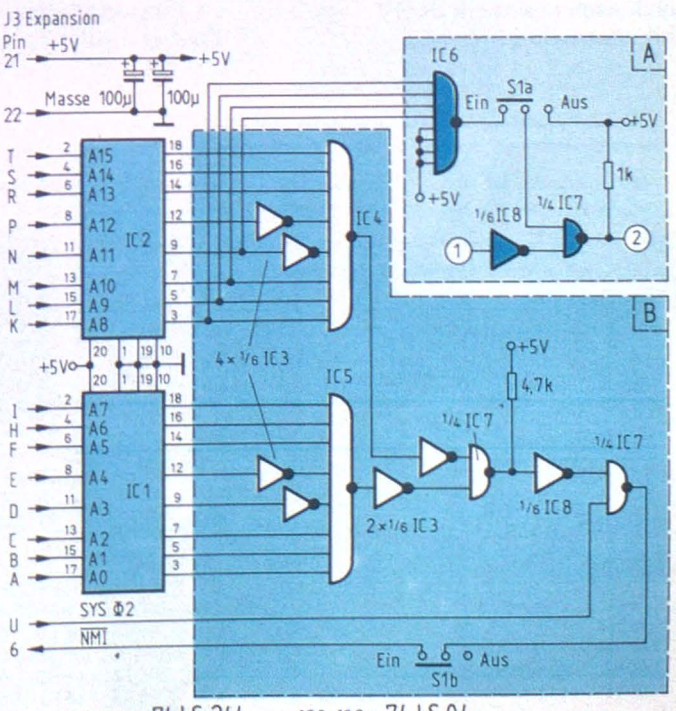


Bild 2. ► Hardware zur Adressenpufferung und -decodierung

IC1, IC2: 74 LS 244 IC3, IC8: 74 LS 04
IC4, IC5, IC6: 74 LS 30 IC7: 74 LS 03 (offener Kollektor)

Programms in eine andere Page muß diese entsprechend geschützt werden. Dafür sind noch Inverter frei. Zum Anschluß dieses Schaltungsteils muß auf der Unterseite der AIM-Platine in der Nähe des Run/Step-Schalters eine Leiterbahn aufgetrennt werden (*Bild 3*).

Das Betriebssystem des AIM ist zwar zum Teil für den Betrieb eines Bildschirms eingerichtet, jedoch wird beim Betätigen der Delete-Taste kein entsprechendes Steuerzeichen ausgegeben. Der Schaltungsteil B dekodiert deshalb die Adresse E7E7, die nach Delete gelesen wird und erzeugt einen nicht maskierbaren Interrupt. Beim Auftreten des NMI prüft die zugehörige Software dessen Ursprung und löscht ein Zeichen oder verzweigt zu den Monitorfunktionen für den STEP-Betrieb. Durch Verwendung des NMI bleiben alle IRQ-Möglichkeiten voll erhalten. Mit dem Schalter S1 kann der normale Betrieb des AIM sichergestellt werden, wenn die TV-Software nicht geladen ist. Die Schaltung findet auf einer ca. 10 x 10 cm² großen Lochrasterplatine Platz. Dort kann auch gleich der Datenbus gepuffert werden.

Software

Nach dem Laden des Programms (*Bild 4*) in die Adressen 0F00 bis 0FFF wird durch Start bei 0F00 der Bildschirm initialisiert. Er läßt sich nun (auch nach Reset) mit der F1-Taste ein- und ausschalten.

Vom Benutzerprogramm kann mit Hilfe der Subroutine TVOUT (0F67) ein ASCII-Zeichen im Akkumulator auf dem Bildschirm ausgegeben werden. Sämtliche CPU-Register bleiben dabei unbeeinflußt.

Wird ein anderes Video-Interface als das beschriebene eingesetzt, so kann eine Anpassung dieser Subroutine erforderlich werden.

Um Zeit und Speicherplatz zu sparen, wurde darauf verzichtet, vor jeder Zeichenausgabe die Position des Display-Pointers abzufragen. Wird dieser geändert, ohne ein entsprechendes Zeichen auszugeben, so bleibt die Position des Bildschirmcursors unbeeinflußt. Dies ist im Normalbetrieb belanglos.

Stichworte zum Inhalt

AIM-65, VIC-1, Interrupt, Step-Modus, Delete-Taste

Basic-Erweiterung für AIM-65 und PC-100

Für die praktisch miteinander identischen Mikrocomputer AIM-65 (Rockwell) und PC-100 (Siemens) liefert die Firma GWK in Herzogenrath eine als Kassette oder PROM erhältliche Erweiterung für den Basic-Interpreter. Sie umfaßt einen auch ohne Basic verwendbaren Dezimal/Hexadezimal-Adressenrechner für Addition und Subtraktion in beiden Zahlensystemen (auch gemischt), eine fest gespeicherte ATN-Routine, die beim AIM bisher so vermißten Befehle für das Öffnen und Schließen von Kassettenfiles, einen Befehl für das Neunumerieren eines Basic-Programms in Zehnerschritten (mit Korrektur aller Sprungadressen!), den vom Editor her bekannten Change-Befehl für einfache Programmkorrekturen und den Befehl „Trace", um ein Programm Zeile für Zeile verfolgen zu können. Ferner ist es auch möglich, Stackpointer-Korrekturen auszuführen, z.B. um von einem Unterprogramm zu einer bestimmten Zeile des Hauptprogramms zu springen. Für Maschinensprache-Spezialisten steht jetzt auch SYS zur Verfügung.

Interessant ist vor allem die Methode, wie die Befehlssatz-Erweiterung durchgeführt wurde. Der Interpreter springt nach dem Befehl „END" indirekt über die Zero-Page-Adresse 0000 zum Warmstart des Basic. Alle erweiterten Befehle beginnen daher mit END, gefolgt von einer Zeichenfolge, die im Erweiterungs-PROM decodiert wird und die Ausführung des gewünschten Befehls veranlaßt. An der Adresse 0000 steht deshalb jetzt ein Sprung zum Decodierprogramm im neuen PROM.

Die Werte von Basic-Variablen können außer auf Kassette auch auf andere „Output Devices" (s. AIM-Handbuch) geschrieben werden. Dabei wird eine ähnliche Methode verwendet, wie im „Basic-Texteditor" in FUNKSCHAU 1980, Heft 1, beschrieben wurde.

Erwähnt sei noch, daß es den Herzogenrathern gelungen ist, den Rockwell'schen (eigentlich von Microsoft stammenden) Basic-Interpreter zu „knacken" – das war die Voraussetzung für die Befehlssatz-Erweiterung. Inzwischen ist von GWK sogar ein kommentiertes Listing des AIM-Basic zu haben.

Fe

```
0F00 A2 02 BD 0E   0F60 EF 20 67 0F   0FC0 8D 25 A4 68
0F04 0F 9D 0C 01   0F64 4C 05 EF 48   0FC4 48 C9 E7 F0
0F08 CA 10 F7 4C   0F68 20 9E EB 29   0FC8 11 C9 0F D0
0F0C 1B E1 4C 11   0F6C 7F 8D 00 A0   0FCC 06 AD 25 A4
0F10 0F A9 61 8D   0F70 48 68 09 80   0FD0 4C 97 E2 AD
0F14 06 A4 A9 0F   0F74 8D 00 A0 C9   0FD4 25 A4 48 4C
0F18 8D 07 A4 8D   0F78 8D F0 0C C9   0FD8 82 E0 AD 07
0F1C 03 A4 A9 B8   0F7C 8C F0 1C 20   0FDC A4 C9 EF F0
0F20 8D 02 A4 A9   0F80 AC 0F 20 AC   0FE0 1B AD 15 A4
0F24 FF 8D 02 A0   0F84 EB 68 60 20   0FE4 C9 14 B0 05
0F28 A9 44 8D 0D   0F88 A0 0F A9 0A   0FE8 A9 57 8D 06
0F2C 01 A9 0D 20   0F8C 8D 00 A0 48   0FEC A4 A9 08 20
0F30 67 0F A9 3C   0F90 68 09 80 8D   0FF0 67 0F A9 20
0F34 20 67 0F A9   0F94 00 A0 20 A0   0FF4 20 67 0F A9
0F38 5B 20 67 0F   0F98 0F F0 E7 20   0FF8 08 20 67 0F
0F3C A9 3E 20 67   0F9C A6 0F F0 E2   0FFC 4C 94 E2 EA
0F40 0F 4C FA E6   0FA0 A9 FF A2 07
0F44 20 F1 E6 A9   0FA4 D0 0A A9 FF
0F48 11 8D 0D 01   0FA8 A2 80 D0 04   FUER ANDERE PAGE FOLGENDE
0F4C A9 05 8D 06   0FAC A9 20 A2 01   ZELLEN AENDERN:
0F50 A4 A9 EF 8D   0FB0 A8 88 D0 FD   0F04  0F10  0F17  0F31
0F54 07 A4 60 48   0FB4 CA D0 F9 60   0F36  0F3B  0F40  0F63
0F58 A9 61 8D 06   0FB8 8D 21 A4 68   0F81  0F89  0F98  0F9D
0F5C A4 68 4C 05   0FBC 8D 20 A4 68   0FCA  0FF1  0FF6  0FFB
```

Bild 4. Software für das Interface. Durch den Start bei 0F00 wird der Bildschirm initialisiert

Rudolf Hofer

TRS-80-Treiberprogramm für Schreibmaschinendrucker

Zur Ansteuerung des in Heft 4/1980 beschriebenen Schreibmaschinendruckers ist ein kleines Decodierprogramm erforderlich. Die grundlegenden Anforderungen wurden bereits erläutert. Das konkrete Programm für den TRS-80 wird im folgenden beschrieben.

Originaltreiber verschieben

Erkennt der BASIC-Interpreter der Maschine einen der Befehle LLIST oder LPRINT, dann springt er auf ein Programm, dessen Anfangsadresse in den RAM-Zellen 4026_{16} (= 16422_{10}, LSB) und 4027_{16} (= 16423_{10}, MSB) steht. Normalerweise ist diese Adresse $058D_{16}$. Es ist die Anfangsadresse des Lineprintertreibers im ROM, der in *Bild 1* ausgedruckt ist. Dieses Programm gibt an zwei Stellen ein ASCII-Zeichen an den Drucker aus, und zwar mit dem Befehl „LD (37E8), A" an den Adressen 05AD und 05BB.

Der Befehl muß beide Male durch einen Unterprogrammaufruf auf das Decodierprogramm ersetzt werden. Das geht allerdings nur im RAM-Bereich; deshalb muß der gesamte Programmabschnitt in das RAM transportiert werden. Es ist empfehlenswert, einen Bereich zu wählen, der sich nur im MSB von der ursprünglichen Lage unterscheidet, also etwa 7D8D...7DD8 bei der 16-K-Version oder B58D...B5D8 bei der 32-K-Ausführung, da man dann nur ein Byte der Treiberadresse ändern muß (4027: 05 → 7D bzw. B5).

Decodierprogramm

Die beiden Unterprogrammsprünge führen zum eigentlichen Decodierprogramm *(Bild 2)*. Es entspricht dem Flußdiagramm in Heft 4. Der erste Teil nimmt mit Hilfe der Tabelle die Umcodierung vor, der letzte schaltet zwischen Groß- und Kleinschreibung um und merkt sich den jeweiligen Zustand in einem Flag (Adresse B6A0).

Die Zuordnung zwischen ASCII- und Druckercode ist in der Tabelle gespeichert. An den geraden Adressen findet man die ASCII-Zeichen, jedem einzelnen folgt unmittelbar der Druckercode – in diesem Fall für die Schreibmaschine Gabriele 2000. Diese Tabelle findet sich in anderer Form ebenfalls in H. 4 wieder, sie beginnt mit ASCII 2E (= Punkt) und endet mit einer Reihe von Sonderzeichen ab Adresse B7D0. Hier werden CR ($0D_{16}$) und LF ($0A_{16}$) als Wagenrücklauf (35 im Druckercode) interpretiert, die Null (30_{16}) wird zum Buchstaben O, das Malzeichen ($2A_{16}$) wird ein kleines x usw. Weitere Zeichenpaare können am Schluß anstelle von 34 34 angefügt werden, z. B. die eckige Klammer oder das Zeichen „↑ ".

Auch jede andere Schreibmaschine kann leicht angepaßt werden, indem man die Tabelle an den entsprechenden Stellen ändert. Ebenso ist man völlig frei, was man anstelle von ASCII-Zeichen drucken will, die nicht auf der Schreibmaschine vorhanden sind. Wird ein Zeichen nicht gefunden, druckt die Maschine einen Leerraum, da als letzter Code 34 in den Akku geladen und ausgegeben wird. Soll die Tabelle verlängert werden, muß für jedes zusätzliche Zeichenpaar der Wert in der Speicherstelle B6B1 um eins erhöht werden.

Laden und verschieben

Das gesamte Programm wird mit Hilfe von TBUG in die angegebenen Speicherzellen eingetippt. Also zuerst der Originaltreiber (Bild 1) mit den beiden Änderungen, dann das Decodierprogramm (Bild 2). Alles zusammen nimmt man dann auf Band auf. Dieses Programm kann jederzeit mit SYSTEM wieder eingelesen werden (der Speicherplatz ist mit MEMORY SIZE gleich nach dem Einschalten zu reservieren) – man darf es aber nicht starten! Um es zu

RSM2S-Ausdruck

```
058D:  79          LD   A,C
058E:  B7          OR   A
058F:  28 40       JR   Z,05D1
0591:  FE 0B       CP   0B
0593:  28 0A       JR   Z,059F
0595:  FE 0C       CP   0C
0597:  20 1B       JR   NZ,05B4
0599:  AF          XOR  A
059A:  DD B6 03    OR   (IX+03)
059D:  28 15       JR   Z,05B4

059F:  DD 7E 03    LD   A,(IX+03)
05A2:  DD 96 04    SUB  (IX+04)
05A5:  47          LD   B,A
05A6:  CD D1 05    CALL 05D1
05A9:  20 FB       JR   NZ,05A6
05AB:  3E 0A       LD   A,0A

05AD:  32 E8 37    LD   (37E8),A
05B0:  10 F4       DJNZ 05A6
05B2:  18 18       JR   05CC
05B4:  F5          PUSH AF
05B5:  CD D1 05    CALL 05D1
05B8:  20 FB       JR   NZ,05B5
05BA:  F1          POP  AF
05BB:  32 E8 37    LD   (37E8),A
05BE:  FE 0D       CP   0D
05C0:  C0          RET  NZ
05C1:  DD 34 04    INC  (IX+04)
05C4:  DD 7E 04    LD   A,(IX+04)
05C7:  DD BE 03    CP   (IX+03)
05CA:  79          LD   A,C
05CB:  C0          RET  NZ
05CC:  DD 36 04 00 LD   (IX+04),00
05D0:  C9          RET
05D1:  3A E8 37    LD   A,(37E8)
05D4:  E6 F0       AND  F0
05D6:  FE 30       CP   30
05D8:  C9          RET
```

Bild 1. Original-Lineprintertreiber im ROM: Die Speicherstellen 05AD...F und 05BB...D sind jeweils durch einen Unterprogrammaufruf des Decodierprogramms zu ersetzen (Maschinencode CD A1 B6 = CALL B6 A1); das ganze Programm wird im RAM untergebracht

aktivieren, muß man noch die Treiberadresse in der Speicherstelle 4027_{16} verändern, und zwar mit POKE 16423, 181 (entspricht $B5_{16} \rightarrow 4027_{16}$). Diese Angaben gelten für die 32-K-Version. Hat man nur 16 K RAM, sind folgende Änderungen vorzunehmen:
Originaltreiber nach 7D8D...D8
7DAD...F: CD A1 7E
7DBB...D: CD A1 7E
Codierprogramm nach 7E A1...7F E7
Jedes Byte B6 an der dritten Stelle nach der Adresse (Bild 2) wird durch 7E ersetzt.

Das Byte B7 an der vierten Stelle (Adr. 7EAF) wird durch 7F ersetzt.
Nach jedem Laden des Programms wird es mit POKE 16423, 125 (entspricht $7D_{16} \rightarrow 4027_{16}$) aktiviert.

Danach kann mit den Befehlen LPRINT und LLIST gearbeitet werden wie mit dem vorgesehenen Drucker von Tandy. Achtung: Das Interface muß angeschlossen sein, sonst nimmt der Rechner keine Eingabe mehr an, da er sich in einer endlosen Schleife befindet! (OUT OF PAPER auf Masse, BUSY nicht dauernd „1").

```
VOR DISPLAY-ROUTINE EINFUEGEN
B680:  F5              PUSH AF
B681:  3A E8 37        LD   A,(37E8)   Status einlesen
B684:  CB 6F           BIT  5,A        Device selected?
B686:  28 09           JR   Z,B691     nein, kein Ausdruck
B688:  CB 7F           BIT  7,A        Busy?
B68A:  20 F5           JR   NZ,B681    ja, warten
B68C:  F1              POP  AF
B68D:  CD A1 B6        CALL B6A1       Codierprogramm
B690:  C9              RET
B691:  F1              POP  AF
B692:  C9              RET
```

Bild 3. Dieses Programm ersetzt das von Bild 1 und kann vor der Displayroutine eines Maschinenprogramms aufgerufen werden (alle Listings wurden mit dem Schreibmaschinendrucker und dem beschriebenen Treiberprogramm erstellt.)

Tips für andere Programme

Für den TRS-80 gibt es verschiedene Maschinenprogramme, für die eine Ausdruckmöglichkeit ebenfalls sehr nützlich wäre, z. B. den Monitor RSM2S, der einen Disassembler enthält und mit dem die abgedruckten Listings erstellt wurden. Da hier keine Druckerroutine enthalten ist, behilft man sich, indem man vor die Displayroutine einen Unterprogrammsprung zu der in

```
UMCODIEREN
B6A1:  F5              PUSH AF
B6A2:  E5              PUSH HL
B6A3:  DD E5           PUSH IX
B6A5:  C5              PUSH BC
B6A6:  21 00 05        LD   HL,0500
B6A9:  CD DF B6        CALL B6DF
B6AC:  DD 21 32 B7     LD   IX,B732
B6B0:  06 5A           LD   B,5A
B6B2:  DD 23           INC  IX
B6B4:  DD 23           INC  IX
B6B6:  00              NOP
B6B7:  00              NOP
B6B8:  00              NOP
B6B9:  05              DEC  B
B6BA:  28 05           JR   Z,B6C1
B6BC:  DD BE 00        CP   (IX+00)
B6BF:  20 F1           JR   NZ,B6B2
B6C1:  DD 7E 01        LD   A,(IX+01)
B6C4:  FE 35           CP   35
B6C6:  20 0B           JR   NZ,B6D3
B6C8:  32 E8 37        LD   (37E8),A
B6CB:  21 00 60        LD   HL,6000
B6CE:  CD DF B6        CALL B6DF
B6D1:  18 03           JR   B6D6
B6D3:  CD F4 B6        CALL B6F4
B6D6:  C1              POP  BC
B6D7:  DD E1           POP  IX
B6D9:  E1              POP  HL
B6DA:  F1              POP  AF
B6DB:  C9              RET

VERZOEGERUNG
B6DF:  2E F0           LD   L,F0
B6E1:  32 00 00        LD   (0000),A
B6E4:  32 00 00        LD   (0000),A
B6E7:  32 00 00        LD   (0000),A
B6EA:  2D              DEC  L
B6EB:  20 F4           JR   NZ,B6E1
B6ED:  25              DEC  H
B6EE:  20 EF           JR   NZ,B6DF
B6F0:  C9              RET

GROSS/KLEIN
B6F4:  CB 77           BIT  6,A
B6F6:  20 1B           JR   NZ,B713
B6F8:  F5              PUSH AF
B6F9:  3A A0 B6        LD   A,(B6A0)
B6FC:  FE 00           CP   00
B6FE:  28 2B           JR   Z,B72B
B700:  3E 36           LD   A,36
B702:  00              NOP
B703:  32 E8 37        LD   (37E8),A
B706:  3E 00           LD   A,00
B708:  32 A0 B6        LD   (B6A0),A
B70B:  21 00 20        LD   HL,2000
B70E:  CD DF B6        CALL B6DF
B711:  18 18           JR   B72B
B713:  F5              PUSH AF
B714:  3A A0 B6        LD   A,(B6A0)
B717:  FE 01           CP   01
B719:  28 10           JR   Z,B72B
B71B:  3E 33           LD   A,33
B71D:  32 E8 37        LD   (37E8),A
B720:  3E 01           LD   A,01
B722:  32 A0 B6        LD   (B6A0),A
B725:  21 00 20        LD   HL,2000
B728:  CD DF B6        CALL B6DF
B72B:  F1              POP  AF
B72C:  E6 3F           AND  3F
B72E:  32 E8 37        LD   (37E8),A
B731:  C9              RET

TABELLE
B734:  2E 03 21 43 2D 04 27 44 2B 08 2F 48 6F 09 4F 49
B744:  6B 0A 4B 4A 6D 0B 4D 4B 2C 0C 3F 4C 6C 0D 4C 4D
B754:  70 0E 50 4E 38 10 40 2E 75 11 55 51 68 12 48 52
B764:  62 13 42 53 6E 14 4E 54 6A 15 4A 55 69 16 49 56
B774:  36 18 28 58 74 19 54 59 66 1A 46 5A 63 1B 43 5B
B784:  76 1C 56 5C 67 1D 47 5D 7A 1E 5A 5E 34 20 25 60
B794:  65 21 45 61 73 22 53 62 79 23 59 63 78 24 58 64
B7A4:  64 25 44 65 72 26 52 66 39 29 24 69 37 2A 29 6A
B7B4:  35 2B 26 6B 33 2C 3D 6C 77 2D 57 6D 61 2E 41 6E
B7C4:  32 30 22 70 31 31 3B 71 71 32 51 72 0D 35 0A 35
B7D4:  3A 68 20 34 30 49 2A 24 23 01 34 34 34 34 34 34
B7E4:  34 34 34 34
```

Bild 2. Decodierprogramm für die 32-K-Version

Bild 3 gezeigten Befehlsfolge einfügt. Das auszugebende Zeichen befindet sich zu diesem Zeitpunkt im Akku. Falls die Leitung „Device selected" am Lineprinterstecker nicht low ist, wird es von der Schreibmaschine ausgedruckt, anderenfalls wird das Zeichen nur auf den Schirm geschrieben. Natürlich muß das Codierprogramm von Bild 2 ebenfalls geladen sein.

Die Displayroutine findet man, indem man nach CALL 0033 (= CD 33 00) sucht. Diese Methode kann man grundsätzlich anwenden, auch bei anderen Programmen. Zu beachten ist, daß sich das auszugebende Zeichen vor der Displayroutine nicht bei jedem Programm im Akku befindet. Oft wird Register C als Zwischenspeicher verwendet – z. B. vom BASIC-Interpreter. *Rudolf Hofer*

Literatur

[1] *Hofer, R.:* Schreibmaschine wird zum Schönschreibdrucker. FUNKSCHAU 1980, Heft 4

[2] *Feichtinger, H.:* 6502-Programm zur Schreibmaschinen-Ansteuerung. FUNKSCHAU 1980, Heft 4

Stichworte zum Inhalt
Z-80, Line Printer, Maschinensprache, TBUG, ASCII, Treiber

Basic-Texteditor-Verbesserungen

Der in Heft 1/1980 der FUNKSCHAU auf Seite 87 abgedruckte Basic-Texteditor konnte in einigen Punkten bedienungsfreundlicher gemacht werden.

Wesentlich ist die Einführung einer schnelleren Sortier-Routine (*Bild 1*). Bis auf das Sortieren eines bereits sortierten Feldes (was allerdings ohnehin unsinnig ist) ergibt sich damit eine erhebliche Zeitersparnis; zum Beispiel dauert das Sortieren von 12 neuen Zeilen in eine Datei mit 62 Zeilen statt 166 Sekunden nur noch 17 Sekunden.

Das Suchen nach einem Stichwort konnte ebenfalls schneller gemacht werden (*Bild 2*). Hierbei werden zunächst – wie das z. B. beim Suchen nach einer Telefonnummer praktisch ist – nur die Wortanfänge nach dem Suchwort durchforstet. Ist dies geschehen, so wird eine Zeile mit Sternchen ausgegeben, danach beginnt erst das Suchen ab dem zweiten Zeichen der Texte. Die Suche läßt sich (ebenso wie das Sortieren) jederzeit mit der Leertaste abbrechen.

Schließlich wurde das „Rangieren" mit den Tasten U (Up) und D (Down) komfortabler gemacht, weil jetzt beim Niederhalten dieser Tasten eine Dauerfunktion möglich ist (*Bild 3*).

Peter Lindner

```
1100 PRINT"■SORTING..... ";:FORI=0TO0:IFA$(I)<>"END"THENNEXT
1110 I=I-1:L=1:N(L)=I+1:M=1
1120 J=N(L):K=M-1:IFJ-M<3GOTO1230
1130 M1=INT((K+J)/2):GETM$:IFM$=" "THENPRINT"SORT
     INTERRUPTED!":RETURN
1140 K=K+1:IFK=JGOTO1190
1150 IFA$(K)=<A$(M1)GOTO1140
1160 J=J-1:IFK=JGOTO1190
1170 IFA$(J)>=A$(M1)GOTO1160
1180 Y$=A$(K):A$(K)=A$(J):A$(J)=Y$:GOTO1140
1190 IFK>=M1THENK=K-1
1200 IFJ=M1GOTO1220
1210 Y$=A$(K):A$(K)=A$(M1):A$(M1)=Y$
1220 L=L+1:N(L)=K:GOTO1120
1230 IFJ-M<2GOTO1260
1240 IFA$(M)<A$(M+1)GOTO1260
1250 Y$=A$(M):A$(M)=A$(M+1):A$(M+1)=Y$
1260 M=N(L)+1:L=L-1:IFL>0GOTO1120
1270 PRINT"       SORTED!■":RETURN
```

Bild 1. Schnelleres Sortieren von Zeilen erlaubt diese Routine. In Zeile 160 muß „180" durch „1100" ersetzt werden

```
270 J=-1:INPUT"■WORD ";B$:K=LEN(B$):PRINT
280 J=J+1:IFA$(J)="END"THENPRINT"**********************************
    *********":J=-1:GOTO310
290 IFB$=LEFT$(A$(J),K)THENPRINTA$(J):GOTO280
300 GOTO280
310 J=J+1:IFA$(J)="END"THENPRINT:RETURN
320 GETM$:IFM$=" "THENPRINT"SEARCH INTERRUPTED!":J=D:RETURN
330 S=LEN(A$(J))-K:IFS<0GOTO310
340 FORI=2TOS+1:M$=MID$(A$(J),I,K):IFM$=B$GOTO360
350 NEXT:GOTO310
360 PRINTA$(J):GOTO310
```

Bild 2. Auch die Stichwortsuche kann erheblich beschleunigt werden, indem zunächst nur die Wortanfänge durchforstet werden

```
120 RESTORE:GOSUB1050
130 FORI=1TO15:READB$:IFB$=M$GOTO150
140 NEXT:GOTO120
150 IFI=15GOTO50
160 ONIGOSUB1100,230,250,270,370,450,500,580,660,760,700,730,740
170 PRINTA$(J)
172 T=TI+1
174 IFPEEK(515)=255ORPEEK(515)<>47ANDPEEK(515)<>61GOTO120
176 IFTI<TGOTO174
178 GOTO160
1050 POKE548,0:WAIT525,1:GETM$:POKE548,1:RETURN
```

Bild 3. Mit wenig Aufwand kann man schließlich eine Dauerfunktion für die Tasten U und D erreichen, um schneller zu einer bestimmten Zeile zu gelangen. In Zeile 160 ist bereits die schnellere Sortier-Routine berücksichtigt. (Vorsicht: Die PEEK-Adressen gelten nur für den PET-2001!)

µC-Nachrichten

Die Firma Sinclair Research (Cambridge) hat in England einen preiswerten und tragbaren Kleincomputer für Heimanwendungen vorgestellt, der ab April geliefert werden soll. Wesentlichstes Merkmal: Der Preis soll unter 100 Pfund Sterling liegen. Das Gerät ist für den Anschluß eines Fernsehgerätes (britische Norm) vorbereitet und heißt ZH-88. Sinclair plant, den größten Teil der Fertigung in die USA zu liefern.

□

Die Aachener Firma GWK wird in ihr 6502-orientiertes Programm nach dem AIM-65 nun auch den Kompaktcomputer PC-100 aufnehmen. Dabei handelt es sich praktisch um einen mit Netzteil, Basic-Interpreter und Gehäuse versehenen AIM-65.

□

In Hainburg hat sich eine Interessengemeinschaft für den Tausch von TI-Rechnerprogrammen (TI-58/59) etabliert. Die Haupt-Interessengebiete sind Elektronik, Mathematik und Spiele. Alle zwei Monate erscheint eine Clubzeitschrift (DM 1,50). Informationen sind gegen Rückporto von Thomas Brettinger, Schillerstraße 13, 6452 Hainburg, zu haben.

Thomas Blaffert

Z-80-Urlader

Eine vielseitige und sichere Methode, sein Betriebssystem in den Computer zu bekommen, ist das Laden des Programms eines anderen Computers vom Tonband. Dazu muß allerdings ein festverdrahtetes Programm vorliegen, welches das Einlesen bewirkt. Besteht dieser Urlader aus nur wenigen Bytes, so kann er einfach als Diodenmatrix eingelötet werden.

Derzeit werden zwar schon eine Reihe fertiger Hobbycomputer angeboten, doch kann sich für jemanden, der seine Freizeit nicht mit Geld aufwiegt, der Aufbau eines µP-Systems aus einzeln gekauften ICs finanziell durchaus noch lohnen. Grundsätzlich stellt sich aber das Problem, dieses selbstgeschaffene Werk überhaupt zum Laufen zu bewegen, da schließlich PROMs mit einem fertigen Betriebssystem fehlen. Die Eigenprogrammierung von PROMs oder EPROMs setzt sowohl entsprechende Geräte als auch ein fertiges Programm voraus. Beides ist bei einem µP-Erstling aber kaum vorhanden.

Der vielseitige Befehlssatz der Z-80-CPU erlaubt es, einen Urlader unter geringfügigem Hardware-Einsatz mit nur 16 Byte Programm aufzubauen. Wegen Doppelbenutzung sind nur 13 Byte fest verdrahtet. Vorausgesetzt wird dabei, daß ein Serieninterface (USART) 8251 und ein FSK-Modulator/Demodulator vorhanden ist (FSK-Modem siehe FUNKSCHAU 14/78, 15/78).

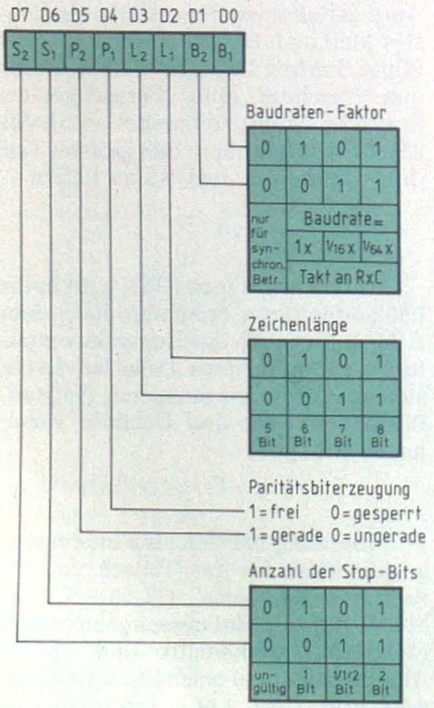


Bild 1. Programmiermöglichkeiten des 8251 bei asynchroner Betriebsweise, erstes Kommandowort

Das Urladeprogramm

Das Programm befindet sich an den Adressen 0000 bis 000F (*Tabelle*). Vor Benutzung des USART 8251 muß dieses mit zwei Kommandos in die richtige Betriebsweise gebracht werden. Der erste Befehl des Programms lädt nun die Adresse dieser Kommandoworte (000E) in Register HL. Durch den folgenden Befehl wird gleichzeitig Register B mit der Anzahl der Kommandoworte (2) sowie das Register C mit der Adresse des USART 8251, Control-Pfad (hier: 05) geladen. Dies sind Vorbereitungen für den Block-Ausgabebefehl OTIR, der das Schreiben beider Kommandoworte in den USART bewirkt. Anschließend ist dieses für den Empfang von asynchronen Daten mit Parity-Bit und 1 Stop-Bit bei 300 Baud eingestellt, sofern am Takteingang RxC des USART eine Taktfrequenz von 4800 Hz anliegt. Eine andere Programmierung ist ohne weiteres möglich (*Bild 1*).

Nachdem das Register C mit der Adresse des Datenpfades USART geladen ist (hier: 04), können über den Blockeingabebefehl INIR Daten in den Speicher geschrieben werden. Man muß sich dazu vergegenwärtigen, daß das Register HL durch den OTIR-Befehl auf die Adresse 0010 gesetzt wurde, ab dieser Adresse also jetzt der Speicher belegt wird. Außerdem steht das Zählregister B auf 00, was bedeutet, daß der INIR-Befehl im Folgenden 256 mal durchlaufen wird. Anschließend wird mit der Ausführung des gerade gelesenen Programms fortgefahren.

Hardware

Die Eingaberate von etwa 10 Byte/s liegt nun weit unter der Geschwindigkeit, mit welcher der INIR-Befehl Bytes lesen würde (ca. 100 kByte/s). Deshalb ist noch eine Zusatzschaltung erforderlich, welche die vollständige Ausführung des Eingabebefehls stoppt, bis ein weiteres Datenbyte empfangen worden ist (*Bild 2*). Das Lesen eines Bytes bewirkt beim USART das Setzen des RxRdy-Signals (Receiver Ready) auf „0". Nach schreiben dieses Bytes in den Speicher springt die CPU wieder an den Anfang des INIR-Befehls, also zur Adresse 000A. Dieser Befehl wird erneut gelesen, wobei die CPU auch die Adresse 000B anspricht. Nun wird das Signal AdrB (Adresse 000B) „0", was zusammen mit $\overline{\text{BLDR}}$ = „0" (Bootstraploader) und RxRdy = „0" auch das $\overline{\text{WAIT}}$-Signal auf „0" setzt. Die CPU wartet jetzt so lange mit der weiteren Ausführung bis RxRdy = „1", also ein neues Datenbyte im USART vorliegt.

Bild 2 zeigt weiterhin die Verdrahtung des Urladers und dessen Anschaltung an das Bus-System. Beim Einschalten der Betriebsspannung wird ein Reset-Signal erzeugt, das den Befehlszähler der CPU auf 0000 setzt und den

Tabelle: Urlade-Programm im Z-80-Maschinencode

```
LOC  OBJ CODE  STMT  SOURCE    STATEMENT
0000 210E00      1             LD    HL,KOMWRT; ADR. DES KOMMANDO-
                 2                               WORTS IN HL
0003 010502      3             LD    BC,0205H ; REG B = 2, REG C = 5
0006 EDB3        4             OTIR           ; KOMMANDOWORT IN 8251
0008 0E04        5             LD    C,04     ; REG C = ADR.8251
                 6                              ; (DATENPFAD)
000A EDB2        7             INIR           ; DATEN IN SPEICHER
000C 1802        8             JR    RUN      ; EINGEL. PROGRAMM
                 9                              ; AUSFUEHREN
000E 7E         10 KOMWRT      DEFB  01111110B1; KOMMANDOWORTE FUER
000F 14         11             DEFB  00010100B ; USART 8251
                12 RUN         EQU   0010H     ; STARTADRESSE DES
                                                 EINGELESENEN
                13                               PROGRAMMS
```

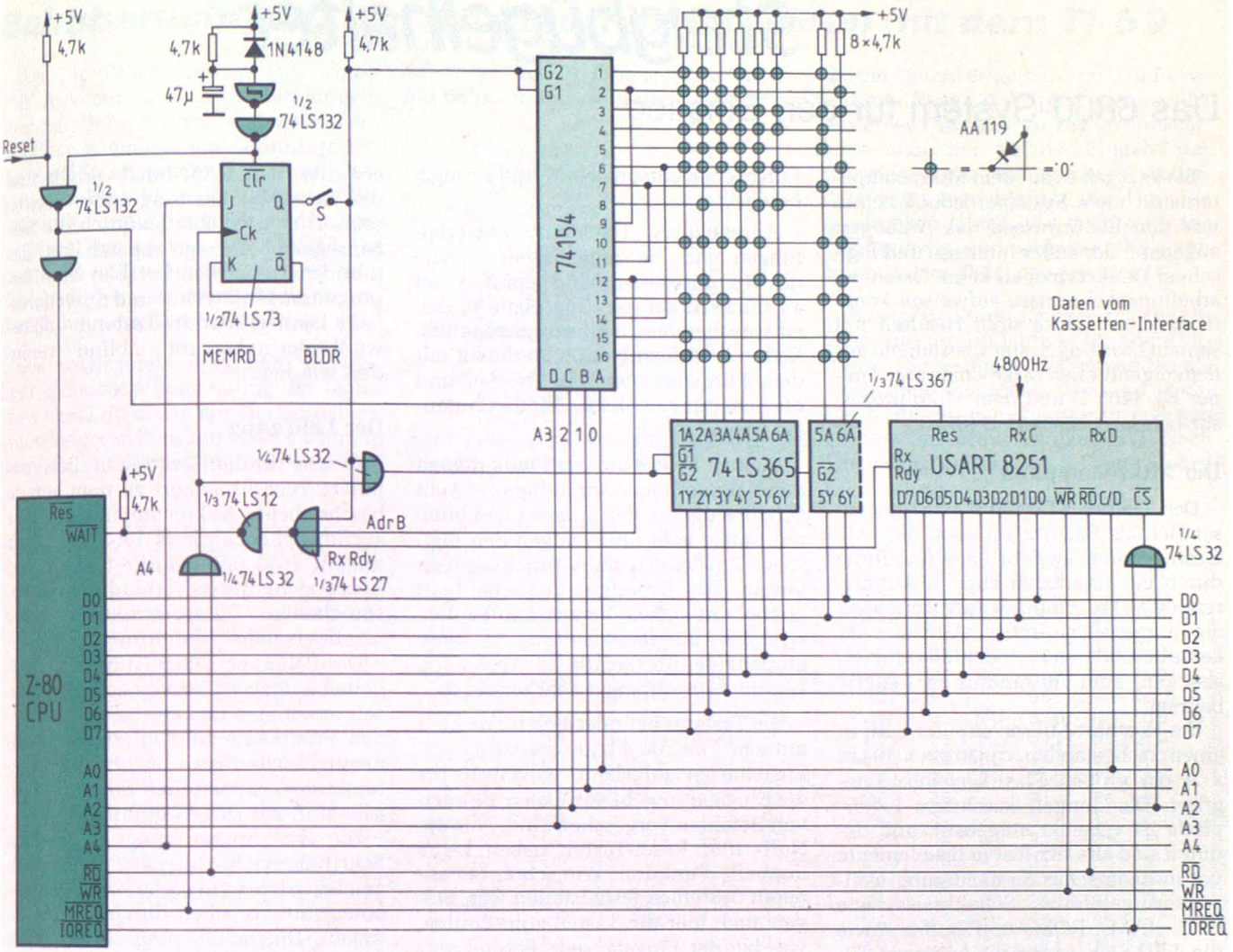


Bild 2. Gesamtschaltung der Urlader-Hardware. Als FSK-Modem können die in den Heften 14 und 15/1978 angegebenen einfachen Schaltungen dienen

USART bereit zum Empfang der Kommandos macht. Außerdem soll jetzt das Ladeprogramm auf dem Datenbus liegen, d.h. Signal $\overline{\text{BLDR}}$ wird „0". Sobald sämtliche 256 Bytes eingelesen sind, wird die Adresse 0010 angesprochen, was zum Setzen des Flipflop auf $\overline{\text{BLDR}}$ = „1" führt. Ab jetzt wird der Urlader, auch nach einem Reset-Signal, nicht mehr angesprochen. Durch den Schalter S läßt sich der Urlader abschalten, wenn man ein (E)PROM mit festem Programm benutzen möchte. Ist S nämlich geöffnet, liegt die Leitung BLDR auf „1" und ist damit ständig inaktiv.

Die Diodenmatrix wird zeilenweise von dem 4-zu-16-Decoderbaustein 74154 angesteuert, die Datenbytes gelangen über den Puffer 74LS365/367 auf den Datenbus. Die Adressensignale A0...A3 wählen einen 74154-Ausgang aus, der auf „0" geht, während alle anderen Ausgänge auf „1" bleiben. Diejenigen Datenbits der angewählten Zeile, an denen sich eine Diode befindet, werden hierdurch ebenfalls auf „0" gelegt. Datenbits ohne Diode werden durch 4,7k-Widerstände auf 5V-Potential gehalten, signalisieren also „1". Über die Eingänge G1, G2 werden beide TTL-Bausteine aktiviert, wenn das Signal $\overline{\text{BLDR}}$ auf „0" liegt. $\overline{\text{BLDR}}$ kann auch verwendet werden um den schließlich auch noch vorhandenen Speicher abzuschalten, solange das Urladeprogramm bearbeitet wird.

Praktischer Gebrauch

Zur Inbetriebnahme des Computers startet man das Band mit dem zu ladenden Programm (dem Betriebssystem). Es muß einen Vorspann von logisch 1 haben (2400 Hz im Kansas-City-Standard). Jetzt wird der Netzschalter des Geräts betätigt. Das Ladeprogramm wird bis zur Adresse 000B ausgeführt, wo die CPU wartet, bis das erste Byte vom Band eingelesen wird. Das Einlesen aller 256 Bytes ist in etwa 10 s erledigt. Anschließend können, wenn das Betriebssystem so weit gediehen ist, Kommandos über eine Tastatur eingegeben werden.

Das Urladeprogramm und die Zusatzschaltung arbeiteten bisher völlig problemlos. Schwierigkeiten gab es nur beim Aufbau eines Kassetten-Interface und naturgemäß bei der Erstellung des ersten Betriebssystems. Letzteres ließ sich mit einer kleinen zusätzlichen Experimentierschaltung für direkten Speicherzugriff [5] ohne Hilfe eines Gastcomputers herstellen.

Literatur

[1] Z-80-CPU, Technical Manual, Datenbuch der Fa. Zilog.
[2] 8080 Microcomputer Peripherals, User's Manual, Datenbuch der Fa. Intel.
[3] FSK-Demodulator, FUNKSCHAU 14/78, S. 698.
[4] FSK-Oszillator, FUNKSCHAU 15/78, S. 742.
[5] Anregungen dazu kamen aus Beiträgen der Reihe „Praxis mit Mikroprozessoren" von Horst Pelka, FUNKSCHAU 16/77, 18/77.

Steigbügelhalter

Das 6800-System für den Einstieg

Bis jetzt gab es auf dem Mikrocomputermarkt viele Systeme; jedoch keines war den Bedürfnissen des Anfängers angepaßt, der außer Interesse und logischem Denkvermögen keine Datenverarbeitungskenntnisse aufweisen konnte. In diese Lücke stößt Heathkit mit seinem Learning System, bestehend aus dem eigentlichen Mikrocomputer Trainer ET-3400 D und dem Mikroprozessor-Lehrkurs EE-3401 D (*Bild*).

Der Mikrocomputer

Der Trainer ET-3400 D wird grundsätzlich als Bausatz geliefert, der vom Benutzer unter ausführlicher Anleitung durch ein Handbuch zum funktionierenden Mikrocomputer aufzubauen ist; ein unproblematischer Aufbau, der keine besonderen Fertigkeiten verlangt, aber auch zum Verständnis der Materie beiträgt.

Der Heathkit-Mikrocomputer wird in einem recht stabilen, ca. 30 cm x 30 cm x 9 cm großen Plastikgehäuse integriert. Die doppelt kaschierte Leiterplatte ist versenkt eingebaut, und dadurch sind alle sichtbaren Bauelemente vor mechanischer Beschädigung weitgehend geschützt. Schaltkreise, wie CPU, RAMs, ROMs und Buffer sowie die LED-Siebensegment-Anzeigen sitzen auf Sockeln. Die sichtbare Seite der Leiterplatte ist weiß lackiert und durch blaue Linien in Felder aufgeteilt, die die Schaltkreise und Bauelemente einer bestimmten Funktion abteilen (z.B. das CPU-Feld, das ROM-Feld usw.); den Funktionen entsprechend sind sie auch beschriftet.

Adressenbus, Datenbus, Steuerleitungen und die zugehörigen Signale des Mikroprozessors sind gepuffert und an Buchsen auf der Frontplatte in den entsprechenden Feldern zugänglich. Dadurch ist man beim Experiment mit dem Mirocomputer sehr flexibel und die Schaltungen bleiben für den Anfänger übersichtlich.

Weitere Felder auf der Platte dienen dem Experimentierfreudigen: Acht Schalter dienen der Eingabe von binären Daten, acht LEDs zeigen den logischen Zustand an den Versuchsaufbauten an. Das Experimentier-Schaltbrett ermöglicht lötfreie Versuchsaufbauten, wie z.B. Interfaceschaltungen. Diese großzügige, übersichtliche Auslegung kommt dem Anfänger sehr entgegen.

Die Tastatur befindet sich im rechten unteren Teil der Platte; sie ist aus 17 Einzeltasten aufgebaut, wovon 15 für die Eingabe sowohl von Daten als auch von Befehlen vorgesehen sind. Nur die Null- und Reset-Tasten haben keine doppelte Funktion. Wie schon bei anderen Systemen festzustellen war, neigen auch hier die Tasten zum Prellen, was bei der Eingabe sehr störend wirken kann.

Über dem im Gehäuse eingebauten Netzteil ist eine LED als Betriebsanzeige und ein Standby-Schalter zu finden. Dieser Schalter wäre eine sinnvolle Einrichtung, wenn er, wie man es erwartet, den RAM-Inhalt erhält und den Rest der Schaltung außer Betrieb setzt. Hier wird aber lediglich die Siebensegment-Anzeige abgeschaltet, das (übrigens recht komfortable) Monitorprogramm läuft weiter und Speicherinhalte können über die Tastatur – ob bewußt oder unbewußt – „blind" verändert werden.

Der Mikrocomputer ist hier nur Mittel zum Zweck: Im Vordergrund steht ein umfangreicher, hervorragend ausgearbeiteter Kurs, mit dem auch Anfängern die Scheu vor komplizierter Technik genommen wird

Der Lehrgang

Gerade für den Einstieg in diese populäre Technik gehört zu dem gerade beschriebenen Mikrocomputer der sehr ausführliche Lehrkurs EE-3401 D. Er umfaßt zwei prallgefüllte Ordner und Klapptafeln, die in Verbindung mit besprochenen Compactcassetten eine „audiovisuelle" Einführung bilden.

Der Lehrgang selbst ist in zehn Einheiten aufgebaut, wovon acht das Wissen aufbereiten und zwei die begleitenden Programm- und Interface-Experimente beschreiben – bis hin zur Tastaturabfrage, zur seriellen Ein- und Ausgabe und zur D/A-Wandlung.

Die Einführung erfolgt in kleinen Schritten; nach Abstecken der jeweiligen Lernziele wird erklärender Text geboten, unterbrochen durch Zwischenfragen, Diagramme und Bilder. Zum Ende jeder Lehreinheit folgt ein Abschlußtest, der die Kontrolle des Lernerfolges bewirkt.

Trotz des sonst hervorragenden (deutschen!) Begleitmaterials läßt sich feststellen, daß noch einige wenige Verbesserungen möglich sind. So gehen z.B. wichtige Regeln und Merksätze im Text unter; einfach könnten sie durch besonderen Druck hervorgehoben werden. Sollten trotz der Ausführlichkeit des Lehrtextes noch Unklarheiten bestehen, so werden diese spätestens bei Durchführung eines Experiments beseitigt.

Zusammenfassung

Das Mikrocomputer Learning System von Heathkit ist für Anfänger ausgelegt. Dabei kann der Autodidakt genauso zum gewünschten Lernzeil kommen, wie der Teilnehmer an speziellen Ausbildungskursen; gerade für die Ausbildung an Schulen dürfte das System seine spezielle Eignung finden. Dazu trägt besonders das auf dem deutschen Markt bisher wohl einmalige Kursmaterial bei.

Hans-Joachim Brückner

Bahnberechnung von Amateurfunksatelliten mit dem TI-59

Das Interesse der VHF- und UHF-Amateure an Funkverbindungen über Satelliten hat in der letzten Zeit erheblich zugenommen. Grundlage für die Satellitenbeobachtungen und den Sprechverkehr über Satelliten ist die genaue Kenntnis der Referenz-Umläufe. Diese werden von der AMSAT [1] und in den Rundsprüchen veröffentlicht, bedürfen aber einer ständig neuen Berechnung der Wochenumläufe.

Das vorliegende Programm ermöglicht die exakte Berechnung der täglichen Umläufe sowie der Umläufe über einen längeren Zeitraum. Dabei können auch die russischen RS-Satelliten mit erfaßt werden [2]. Eingegeben werden der Bahnversatz sowie die Umlaufzeit, die man leicht erhalten kann [1] oder errechnen kann [3]. Wird der Ausdruck nur eines Umlaufs pro Tag gewünscht, kann man dies mit der Eingabe des Crossingsektors ermöglichen. Außerdem kann man hiermit nur Umläufe ausdrucken lassen, die für das eigene QTH günstig sind. Dies kann man vorher mit dem Oscalator ermitteln [1].

1613371731.	11
1724312213.	12
1417000000.	13
3317352432.	14
1617003763.	15
22351316.	16
46172437.	17
1331211331.	18
2200370063.	19
3617263732.	20
3500423231.	21
1424360000.	22
4040404040.	23
7575757575.	24

Bild 2. Belegung der Datenspeicher im TI-59; die hier nicht aufgeführten sind Null

Ausgedruckt werden die Crossings in Grad und die Umläufe in Stunden/Minuten. Die weiteren Stellen werden als Dezimalen angegeben. Sie sind wichtig, wenn die Berechnung für mehrere Monate im voraus erfolgen soll. Nach Überschreiten der 24-h-Grenze wird

```
000  00   0   050  69 DP   100  99 PRT  150  58 FIX  200  22 INV  250  11  A   302  43 RCL
001  42 STO   051  04  04   101  69 DP   151  43 RCL  201  88 DMS  251  76 LBL  303  24  24
002  08  08   052  91 R/S  102  02  02   152  03  03  202  58 FIX  252  16 A'   304  71 SBR
003  01  1    053  42 STO  103  69 DP   153  71 SBR  203  02  02  253  32 X:T  305  96 WRT
004  01  1    054  02  02  104  05  05   154  22 INV  204  52 EE   254  43 RCL  306  00  0
005  42 STO   055  69 DP   105  69 DP   155  42 STO  205  22 INV  255  07  07  307  42 STO
006  00  00   056  06  06  106  00  00   156  05  05  206  52 EE   256  77  GE  308  08  08
007  69 DP   057  69 DP   107  91 R/S  157  22 INV  207  22 INV  257  17 B'   309  43 RCL
008  00  00   058  00  00  108  42 STO  158  44 SUM  208  58 FIX  258  61 GTO  310  03  03
009  71 SBR  059  71 SBR  109  06  06   159  03  03  209  85  +   259  11  A   311  61 GTO
010  99 PRT  060  99 PRT  110  99 PRT  160  43 RCL  210  53  (   260  76 LBL  312  02  02
011  69 DP   061  69 DP   111  71 SBR  161  03  03  211  43 RCL  261  24 CE   313  17  17
012  01  01   062  01  01  112  99 PRT  162  88 DMS  212  05  05  262  75  -   314  76 LBL
013  71 SBR  063  71 SBR  113  69 DP   163  42 STO  213  54  )   263  02  2   315  13  C
014  99 PRT  064  99 PRT  114  01  01   164  03  03  214  95  =   264  04  4   316  75  -
015  69 DP   065  69 DP   115  69 DP   165  43 RCL  215  42 STO  265  95  =   317  03  3
016  02  02   066  02  02  116  05  05   166  01  01  216  03  03  266  42 STO  318  06  6
017  71 SBR  067  43 RCL  117  69 DP   167  71 SBR  217  32 X:T  267  03  03  319  00  0
018  99 PRT  068  16  16  118  00  00   168  22 INV  218  02  2   268  43 RCL  320  95  =
019  69 DP   069  69 DP   119  91 R/S  169  44 SUM  219  04  4   269  23  23  321  42 STO
020  03  03   070  03  03  120  42 STO  170  05  05  220  32 X:T  270  71 SBR  322  04  04
021  69 DP   071  69 DP   121  07  07   171  75  -   221  77  GE  271  96 WRT  323  61 GTO
022  05  05   072  05  05  122  99 PRT  172  53  (   222  24 CE   272  69 DP   324  14  D
023  71 SBR  073  69 DP   123  61 GTO  173  43 RCL  223  61 GTO  273  28  28  325  76 LBL
024  99 PRT  074  00  00  124  11  A    174  01  01  224  25 CLR  274  07  7   326  17 B'
025  69 DP   075  43 RCL  125  76 LBL  175  54  )   225  76 LBL  275  32 X:T  327  98 ADV
026  01  01   076  17  17  126  99 PRT  176  95  =   226  25 CLR  276  43 RCL  328  43 RCL
027  71 SBR  077  69 DP   127  73 RC*  177  94 +/-  227  43 RCL  277  08  08  329  17  17
028  99 PRT  078  04  04  128  00  00   178  88 DMS  228  02  02  278  77  GE  330  69 DP
029  69 DP   079  91 R/S  129  69 DP   179  44 SUM  229  44 SUM  279  10 E'   331  04  04
030  02  02   080  42 STO  130  20  20   180  03  03  230  04  04  280  43 RCL  332  43 RCL
031  71 SBR  081  03  03  131  92 RTN  181  43 RCL  231  03  3   281  03  03  333  03  03
032  99 PRT  082  69 DP   132  76 LBL  182  05  05  232  06  6   282  61 GTO  334  69 DP
033  69 DP   083  06  06  133  22 INV  183  71 SBR  233  00  0   283  02  02  335  06  06
034  03  03   084  43 RCL  134  65  ×   184  22 INV  234  32 X:T  284  17  17  336  43 RCL
035  69 DP   085  16  16  135  01  1    185  75  -   235  43 RCL  285  76 LBL  337  16  16
036  05  05   086  69 DP   136  00  0    186  53  (   236  04  04  286  96 WRT  338  69 DP
037  69 DP   087  04  04  137  00  0    187  43 RCL  237  77  GE  287  69 DP   339  04  04
038  00  00   088  91 R/S  138  95  =   188  05  05  238  13  C   288  01  01  340  43 RCL
039  71 SBR  089  42 STO  139  22 INV  189  54  )   239  61 GTO  289  69 DP   341  04  04
040  99 PRT  090  04  04  140  59 INT  190  95  =   240  14  D   290  02  02  342  69 DP
041  69 DP   091  69 DP   141  55  ÷   191  94 +/-  241  76 LBL  291  69 DP   343  06  06
042  04  04   092  06  06  142  01  1    192  22 INV  242  14  D   292  03  03  344  69 DP
043  91 R/S  093  69 DP   143  00  0    193  44 SUM  243  32 X:T  293  69 DP   345  00  00
044  42 STO  094  00  00  144  00  0    194  05  05  244  43 RCL  294  04  04  346  01  1
045  01  01   095  71 SBR  145  95  =   195  88 DMS  245  06  06  295  69 DP   347  01  1
046  69 DP   096  99 PRT  146  92 RTN  196  44 SUM  246  32 X:T  296  05  05  348  42 STO
047  06  06   097  69 DP   147  76 LBL  197  03  03  247  77  GE  297  69 DP   349  00  00
048  43 RCL  098  01  01  148  11  A    198  43 RCL  248  16 A'   298  00  00  350  61 GTO
049  16  16   099  71 SBR  149  22 INV  199  03  03  249  61 GTO  299  92 RTN  351  11  A
                                                                  300  76 LBL  352  00  0
                                                                  301  10 E'   353  00  0
```

Bild 1. Nach dem Programmstart mit R/S wird (a) die Umlaufzeit in Stunden und Minuten sowie der Bahnversatz in Grad, (b) das Referenz-Crossing mit Zeit und Grad und (c) der gewünschte Crossing-Sektor in Grad eingegeben. Die Datenausgabe erfolgt in der vorliegenden Form mit dem Drucker

```
DATENEINGABE
PERIODE T/ GRAD
   1.4323141     ZEIT
    25.80815     GRAD
ANFANG T / GRAD
      1.4165     ZEIT
        65.2     GRAD
SEKTOR VON
           0.
BIS
         360.

    3.2488141     ZEIT
     91.00815     GRAD

    5.0811282     ZEIT
     116.8163     GRAD

    6.5134423     ZEIT
     142.62445    GRAD

    8.3457564     ZEIT
     168.4326     GRAD

   10.1780705     ZEIT
     194.24075    GRAD

   12.0103846     ZEIT
     220.0489     GRAD

   13.4426987     ZEIT
     245.85705    GRAD

   15.2750128     ZEIT
     271.6652     GRAD

   17.1073269     ZEIT
     297.47335    GRAD

    18.539641     ZEIT
     323.2815     GRAD

   20.3719551     ZEIT
     349.08965    GRAD

   22.2042692     ZEIT
        14.8978   GRAD
.....................


DATENEINGABE
PERIODE T/ GRAD
   1.4323141     ZEIT
    25.80815     GRAD
ANFANG T / GRAD
      1.4165     ZEIT
        65.2     GRAD
SEKTOR VON
         330.
BIS
         360.

   20.3719551     ZEIT
     349.08965    GRAD
a).....................

   20.4243525     ZEIT
     350.40375    GRAD
.....................

   20.4767499     ZEIT
     351.71785    GRAD
.....................

   20.5291473     ZEIT
     353.03195    GRAD
.....................

   20.5815447     ZEIT
     354.34605    GRAD
.....................

   21.0339421     ZEIT
     355.66015    GRAD
.....................

   19.2540254     ZEIT
      331.1661    GRAD
   21.0863395     ZEIT
     356.97425    GRAD
.....................
^^^^^^^^^^^^^^^^^^^^^
```

```
DATENEINGABE
PERIODE T/ GRAD
   1.4323141     ZEIT
    25.80815     GRAD
ANFANG T / GRAD
      1.4165     ZEIT
        65.2     GRAD
SEKTOR VON
           60.
BIS
           70.
.....................

   1.4688974     ZEIT
     66.5141     GRAD
.....................
```

```
   1.5212948     ZEIT
     67.8282     GRAD
.....................


   1.5736922     ZEIT
     69.1423     GRAD
.....................
.....................
.....................
.....................
.....................
^^^^^^^^^^^^^^^^^^^^^
```

Bild 3. Beispielausdrucke für den Satelliten Oscar-7. Links die Umläufe eines Tages, in der Mitte für eine Woche im Bereich 330...360° mit a = Tagesgrenze und b = Wochengrenze; rechts ebenfalls für eine Woche, allerdings bezeichnet „1" einige Tages, an denen im vorgewählten Sektor kein Crossing stattfindet

eine Markierung ausgedruckt, eine weitere nach Ablauf einer Woche. Das Programm selbst kann leicht an Hand der Auflistung (*Bild 1*) einprogrammiert werden. *Bild 2* zeigt die Speicherbelegung, *Bild 3* einen Beispielausdruck.

P. Janiesch, W. Veltrup

Literatur

[1] AMSAT-DL-Informationsdienst, Alexander Schoening, 1 Berlin 28, Ludolfingerweg 52.

[2] *Schoening, A.*: Die RS-Satelliten der UdSSR. cq-DL 2/79.

[3] *Raddatz, J.*: Bahnversatz von Erdsatelliten. cq-DL 3/79.

PET berechnet Schwingkreise

Das im *Bild* aufgelistete Basic-Programm erklärt sich weitgehend selbst. Einige typische PET-Basic-Eigenheiten sollen trotzdem erwähnt werden:

Nach dem PRINT-Befehl braucht nur ein Anführungszeichen zu stehen; das zweite, das dem auszugebenden Text folgt, kann weggelassen werden, wenn es das letzte Zeichen der Zeile wäre.

Statt PRINT„Text";A kann man auch schreiben PRINT„Text"A. Ein Trennungszeichen zwischen Text und Variablennamen kann also entfallen.

Lothar Miedel

```
10 REM *** SCHWINGKREISBERECHNUNGEN
12 REM *** CR BY LOTHAR MIEDEL
14 PRINT"S
16 PRINT"SCHWINGKREISBERECHNUNGEN MIT DEM                    PET 2001":PRINT
18 PRINT"DIESES PROGRAMM ERMOEGLICHT DIE BE-     RECHNUNG VON VERSCHIEDENEN
20 PRINT"SCHWINGKREISWERTEN.":PRINT"ES BRAUCHT NUR   DIE ENTSPRECHENDE ZAHL
22 PRINT"EINGEGEBEN WERDEN, DANN WERDEN DIE ZUR  BERECHNUNG ERFORDERLICHEN
24 PRINT"ANGABEN ABGEFRAGT UND DANN DIE BE-      RECHNUNG DURCHGEFUEHRT!
26 POKE59468,14
28 PRINT:PRINT"RSCHWINGKREISBERECHNUNG
30 PRINT"1.FREQUENZBERECHNUNG
32 PRINT"2.INDUKTIVITAETSBERECHNUNG
34 PRINT"3.KAPAZITAETSBERECHNUNG
36 PRINT"4.BERECHNUNG ABSTIMMBEREICH
38 GETA$:IFA$=""THEN38
40 IFA$="1"THEN52
42 IFA$="2"THEN62
44 IFA$="3"THEN72
46 IFA$="4"THEN82
48 IFA$<>"1"ANDA$<>"2"ANDA$<>"3"ANDA$<>"4"THENPRINT"NUR 1 BIS 4":GOTO26
50 END
52 PRINT:PRINT:INPUT"INDUKTIVITAET IN uH";L
54 INPUT"KAPAZITAET IN pF";K
56 F=159/SQR(L*K)
58 PRINT"FREQUENZ IN MHz:"F
60 PRINT:PRINT:GOTO28
62 PRINT:PRINT:INPUT"FREQUENZ IN MHz";F
64 INPUT"KAPAZITAET IN pF";K
66 L=25330/((F↑2)*K)
68 PRINT"INDUKTIVITAET IN uH = "L
70 PRINT:PRINT:GOTO28
72 PRINT:PRINT:INPUT"FREQUENZ IN MHz";F
74 INPUT"INDUKTIVITAET IN uH ";L
76 C=25330/((F↑2)*L)
78 PRINT"KAPAZITAET IN pF = "C
80 GOTO28
82 PRINT:PRINT:PRINT"Ca=ANFANGSKAPAZITAET DES DREHKOS
84 PRINT"Ce=ENDKAPAZITAET DES DREHKOS
86 PRINT"fo=OBERE GRENZFREQUENZ ERREICHT MIT Ca
88 PRINT"fu=UNTERE GRENZFREQUENZ ERREICHT MIT Ce
90 PRINT:INPUT"ANFANGSKAPAZITAET Ca";CA
92 INPUT"ENDKAPAZITAET Ce";CE
94 INPUT"UNTERE GRENZFREQUENZ";FU
96 FO=FU*SQR(CE/CA)
98 PRINT"OBERE GRENZFREQUENZ = "FO
100 GOTO28
```

Ing. (grad.) Hartmut Müller

So funktioniert ein Mikroprozessor

Oft genug wurde beschrieben, wie ein Mikroprozessor extern beschaltet werden muß, um einen Mikrocomputer aufzubauen. Die internen Vorgänge im Prozessor hatten dabei stets etwas Geheimnisvolles an sich. Der folgende Beitrag beschreibt die Funktionsweise von Mikro- und Bit-Slice-Prozessoren und vergleicht sie mit einer Mikroprogramm-Steuereinheit.

Was ist eine Mikroprogramm-Steuerung?

Betrachten wir uns einmal eine einfache Mikroprogramm-Steuereinheit (*Bild 1*). Sie setzt sich aus den drei Grundbausteinen Programmzähler, Mikroprogrammspeicher und Befehlsdecoder zusammen. Das Ausgangswort des Programmzählers ist die Speicheradresse für den Mikroprogrammspeicher. Je nach angewähltem Speicherplatz steht am Ausgang des Speichers ein Wort mit einer bestimmten Bitstruktur (Mikrobefehlswort). Ein Teil dieses Wortes steuert einen Befehlsausgabebaustein, der andere Teil wird entweder als Datenwort oder als Sprungadresse für den Programmzähler benutzt.

Die Steuereinheit ist in dieser Konfiguration nicht in der Lage, auf externe Signale des zu steuernden Gerätes einzugehen. Bei den häufigsten Steuerungsproblemen ist es jedoch notwendig, daß die Steuereinheit auf externe Signale reagiert oder eine Steuerung nur durch externe Signale möglich ist.

Bedingte Sprünge

Damit die Steuereinheit auf externe Signale reagieren und somit ein bestimmtes Signal den Steuerungsablauf ändern kann, muß ein Baustein eingesetzt werden, der aus einer Anzahl externer Signale ein einzelnes Signal nach seinem logischen Pegel befragt. Der jeweilige Zustand (L- oder H-Pegel) des betreffenden Eingangssignals soll den Programm- und somit den Steuerungsablauf ändern, indem er den Programmzähler auf eine im Mikroprogrammspeicher abgelegte Adresse (Sprungadresse) setzt. Das Programm wird dann an der neuen Stelle fortgesetzt.

Die Aufgabe der externen Signalauswahl und Programmzählersteuerung übernimmt ein Multiplexer (MUX), dargestellt in *Bild 2*. Mit dieser Schaltungserweiterung ist zu dem aus einem früheren Artikel [1] bekannten Befehl JMP (*jump* = direkter Sprung auf eine im Mikroprogramm angegebene Adresse) ein weiterer Befehl hinzugekommen: der bedingte Sprung JMPC (*jump on condition*). Die Befehle werden durch die Bitstruktur der MUX-Steuerleitung (MUX-Adresse) definiert; dargestellt in *Tabelle 1*.

Bedingte Ausgabe und Befehlskombination

Weitere Möglichkeiten, um die Steuereinheit zu verbessern sind:

1. Verknüpfung der Eingangssignale mit den Steuerungssignalen; der Mikrobefehl bestimmt, ob ein bestimmtes Eingangssignal einen Steuerungsbefehl (Ausgabe) bewirkt.
2. Den Steuerungsbefehl (Punkt 1) gleichzeitig mit einer Sprungroutine zu koppeln.
3. Unabhängigkeit vom Pegel des Eingangssignals; der Mikrobefehl bestimmt, ob ein H- oder L-Pegel am Eingang eine Reaktion bewirkt.

Um diese Operationen ausführen zu können, wird ein Logikbaustein benötigt, der den MUX mit dem Programmzähler und dem Ausgabebaustein (D-MUX) verknüpft. Dieser Logikbaustein steuert den Programmzähler über dessen Set-Eingang und den D-MUX über den Enable-Eingang. Der Befehlsvorrat dieser Ausbaustufe (*Bild 3*) zeigt *Tabelle 2*.

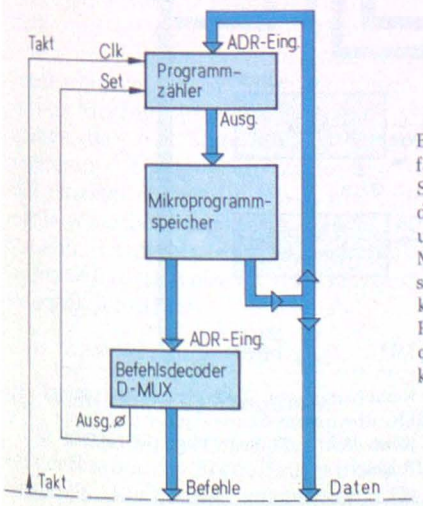


Bild 1. So sieht eine einfache Mikroprogramm-Steuerung aus. Es handelt sich dabei sozusagen um einen abgemagerten Mikrocomputer, der mit sehr hoher Geschwindigkeit arbeiten kann. Der Programmzähler liefert die Adresse für den Mikroprogrammspeicher

Tabelle 1:
Bitstruktur der Befehle auf der MUX-Steuerleitung

MUX-Adresse Dezimal	MUX-Adresse BCD	Befehl	Kurzzeichen des Befehles
0	000	keine Sprungoperation	NOP (no operation)
1	001	direkter Sprung	JMP (jump)
2	010	indirekter Sprung, wenn am betreffenden Eingang L-Pegel	JMPC 2 (jump on condition 2)
3	011	"	"
.	.	↓	↓
.	.		
.	↓		

Die MUX-Adresse wählt den betreffenden Signaleingang aus.

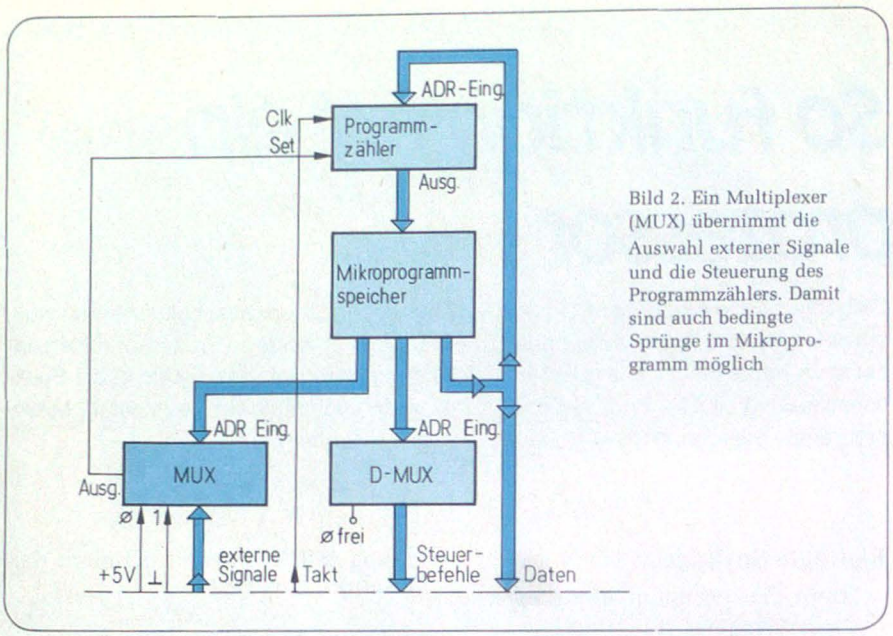


Bild 2. Ein Multiplexer (MUX) übernimmt die Auswahl externer Signale und die Steuerung des Programmzählers. Damit sind auch bedingte Sprünge im Mikroprogramm möglich

Um einen ungestörten zeitlichen Ablauf zu gewährleisten, werden die Steuerungssignale des D-MUX in einem Speicher zwischengespeichert. Aus dem zeitlichen Ablauf innerhalb der Steuereinheit (*Bild 4*) ist zu erkennen, warum der D-MUX-Speicher notwendig ist:

Wenn sich beim nächstfolgenden Programmabschnitt die D-MUX-Adresse ändert, so liegt noch für eine kurze Zeit die alte Logikentscheidung am Enable-Eingang des D-MUX. Das bedeutet, daß der Steuerbefehl für den bereits neuen Programmschritt falsch sein kann.

Unterprogramme

Bei Steuerungsabläufen kann es vorkommen, daß häufig eine bestimmte Routine durchlaufen werden muß. Für diesen Fall ist es sinnvoll, daß dieser Teil als Unterprogramm benannt und vom Hauptprogramm getrennt wird. Bei Bedarf wird dieses Unterprogramm vom Hauptprogramm aufgerufen und abgearbeitet. Nach Beenden des Unterprogramms kehrt das Programm selbständig in das Hauptprogramm zurück.

Die erweiterte Hardware der hier vorgestellten Steuereinheit erstreckt sich für diesen Softwarezusatz auf einen Zählerbaustein (Unterprogrammzähler) und einen BUS-Selektor (*Bild 5*). Der Unterprogrammzähler wird beim Befehl „Unterprogramm" mit der Absprungadresse des Hauptprogramms geladen und im ersten Unterprogrammschritt inkrementiert. Dieser Schritt ist notwendig, damit das Programm beim Befehl „Unterprogramm Ende" auf die im Hauptprogramm nächste Programmzeile springt. Ansonsten würde das Programm wieder den gleichen Befehl „Unterprogramm" ausführen und für immer in dieser Schleife verbleiben (*Bild 6*). Der Befehl „Unterprogramm Ende" schaltet gleichzeitig mit Hilfe des BUS-Selektors den Adreßbus für den Programmzähler vom Mikroprogrammspeicher auf den Unterprogrammzähler (Rücksprungadresse) um. Den Mikrobefehl für die Erweiterung „Unterprogramm" zeigt *Tabelle 3*.

Ein kleines Programmbeispiel

Mit der in Bild 5 dargestellten Ausbaustufe existiert nun eine relativ leistungsfähige Steuereinheit. Das folgende Beispiel soll zeigen, wie ein Mikroprogramm für eine solche Steuereinheit erstellt wird. Wegen des Aufwandes ist bei diesem Beispiel jedoch darauf verzichtet worden, die Vorteile (z.B. hohe Geschwindigkeit) einer solchen Mikroprogramm-Steuerung gegenüber einer Mikroprozessorsteuerung aufzuzeigen. Es zeigt aber, daß sich bereits eine solche Steuereinheit allein – mit Standard-ICs aufgebaut – für viele Zwecke eignet.

Das Problem: Die Steuereinheit soll eine Tür vollautomatisch öffnen oder schließen sowie bei Bedarf die Beleuchtung im Raum und im Vorhof ein- oder ausschalten. Zur Steuerungsrealisierung müssen folgende Kriterien abgefragt werden:

1. Kommt oder geht eine Person?
2. Ist die Tür geöffnet oder geschlossen?
3. Befindet sich eine Person im Vorhof?

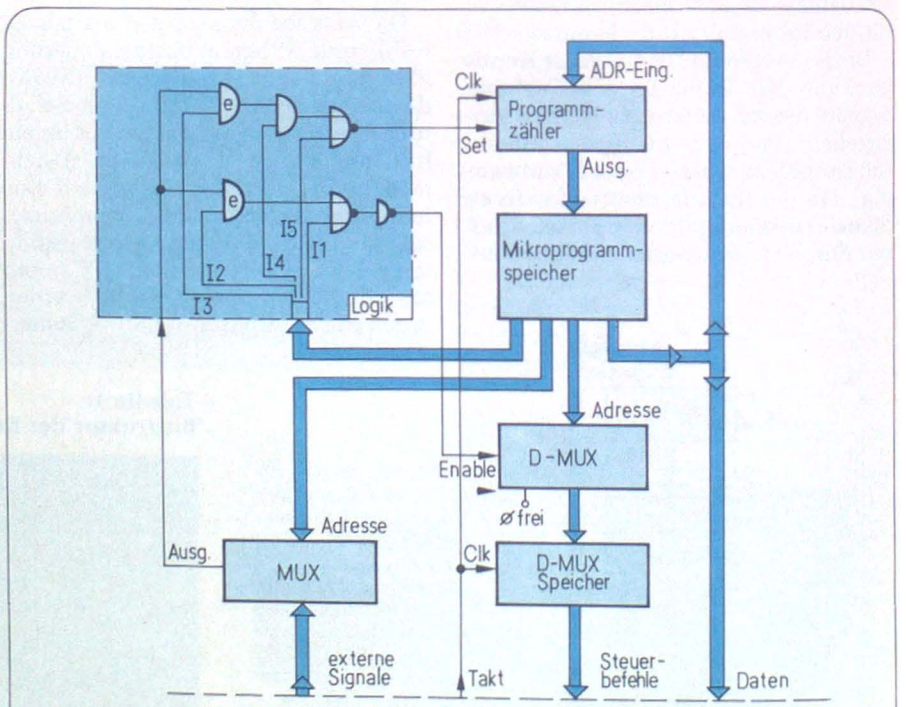


Bild 3. Für die bedingte Ausgabe und die Kombination von Befehlen ist noch eine kleine Erweiterung nötig. Der Programmzähler übernimmt während einer positiven Taktflanke die Adresse vom ADR-Eingang, wenn „Set" auf L-Pegel liegt. Der D-MUX sperrt seine Ausgänge, wenn „Enable" auf L geht

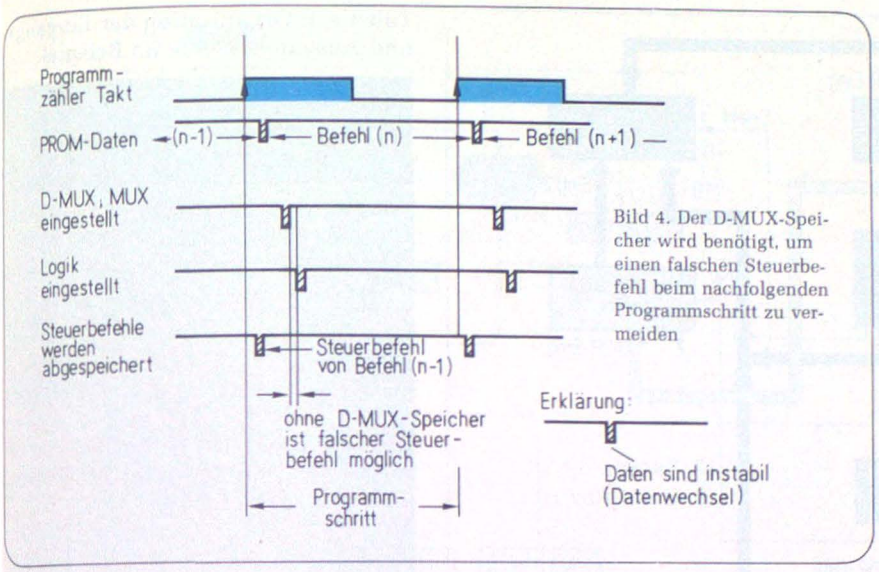


Bild 4. Der D-MUX-Speicher wird benötigt, um einen falschen Steuerbefehl beim nachfolgenden Programmschritt zu vermeiden

4. Ist die Beleuchtung außen notwendig?
5. Befindet sich eine Person im Raum?
6. Ist die Beleuchtung innen notwendig?

Die Steuereinheit gibt je nach Informationsstand folgende Steuersignale an die zu steuernden Geräte ab:

1. Türe öffnen
2. Türe schließen
3. Beleuchtung innen einschalten
4. Beleuchtung innen ausschalten
5. Beleuchtung außen einschalten
6. Beleuchtung außen ausschalten

Zur Programmerstellung wird ein Programmablaufplan benötigt, wie er in Bild 7 dargestellt ist.

Nun müssen den einzelnen Eingangs- und Ausgangssignalen bestimmte Adressen des MUX und D-MUX zugeordnet werden. Die Zusammenstellung, die willkürlich vorgenommen wurde, zeigt Tabelle 4.

Mit dieser letzten Vorarbeit zur Programmerstellung kann das Programm mit Hilfe des Programmablaufplanes (Bild 7), der Befehle (Tabelle 3) und der MUX und D-MUX Organisation (*Tabelle 4*) geschrieben werden. Nach dem Durcharbeiten des in *Tabelle 5* dargestellten Mikroprogramms wird man erkennen, daß das Erstellen des Mikroprogramms (bezogen auf Zeitaufwand und Einsatz) durch die etwas umfangreiche Vorarbeit erleichtert wird. Die Struktur (Aufbau) des Mikrobefehls wurde willkürlich gewählt, sie läßt sich jederzeit verändern.

Die Wahl der Taktfrequenz

Die kleinste Taktperiode setzt sich aus der Summe der Signallaufzeiten der einzelnen Bausteine zusammen. Da sich mehrere Signalwege nach dem Mikroprogrammspeicher ergeben, muß für jeden Signalweg die entsprechende Laufzeit bestimmt werden. Für die Festlegung der kleinsten Taktperiode wird nur der zeitlich längste Signalweg benötigt. So setzt sich die minimal mögliche Periodendauer (T_{CLKmin}) folgendermaßen zusammen:

$$T_{CLKmin} = tv_{Pr.-Z.} + tv_{Pr.-Sp.} + tv_{MUX} + tv_{Log.} + t_{set\ up\ Pr.-Z.}$$

Dabei sind:

$tv_{Pr.-Z.}$ = Laufzeit clock zu den-Ausgängen des Programmzählers;

$tv_{Pr.-Sp.}$ = Laufzeit des Mikroprogrammspeichers vom Adreßeingang zu seinen Ausgängen;

tv_{MUX} = Laufzeit des MUX vom Adreßeingang zu seinem Ausgang;

$tv_{Log.}$ = Laufzeit vom Logikeingang zum Ausgang;

$t_{set\ up\ Pr.-Z.}$ = Set-up-Schaltzeit des Programmzählers.

Bei Verwendung von Schottky-Bausteinen läßt sich für die in Bild 5 dargestellte Schaltung eine maximale Taktfrequenz von 10 MHz erzielen. Durch Verwendung von schnelleren Bausteinen (ECL) läßt sich die Taktfrequenz noch erhöhen.

Tabelle 2: Befehlsvorrat einschließlich der bedingten Sprünge

I_1	I_2	I_3	I_4	I_5	Mikrobefehl	Bemerkung	Kurzzeichen
X	X	Z	L	L	kein Sprung		NOP
X	X	Z	Z	H	direkter Sprung		JMP
X	X	L	H	L	indirekter Sprung bei A	A	JMPC
X	X	H	H	L	indirekter Sprung bei B	B	JMPNC (jump if not condition)
H	Z	X	X	X	keine Ausgabe	C	NOP
H	Z	X	X	X	direkte Ausgabe		OUT (out)
L	L	X	X	X	indirekte Ausgabe bei A	A	OUT C (Out on condition)
L	H	X	X	X	indirekte Ausgabe bei B	B	OUTNC (Out on not condition)

X = für die Betrachtung des betreffenden Befehls uninteressant (Befehlskombination möglich!)
Z = H- oder L-Pegel möglich
Bemerkung:
A = Wenn H-Pegel am MUX-Eingang, dann Sprung bzw. OUT
B = Wenn L-Pegel am MUX-Eingang, dann Sprung bzw. OUT
C = D-MUX-Adresse auf 0 stellen

Tabelle 3: Mikrobefehlsfolge bei Unterprogrammsprüngen

I_6	I_7	I_8	Mikrobefehl	Kurzzeichen
L	H	L	BUS-Selekt schaltet Speicheradresse auf Programmzähler	NOP
L	L	L	Unterprogrammzähler wird mit Programmzählerinhalt geladen	JSUB
L	H	H	Inkrementiere Unterprogrammzähler	INKR
H	H	L	BUS-Selekt schaltet Unterprogrammzähler auf Programmzähler	JRET

Erklärung:
JSUB (jump on subroutine) Sprung ins Unterprogramm
INKR (Inkrement) addiere 1
JRET (jump return) Sprung zurück ins Hauptprogramm

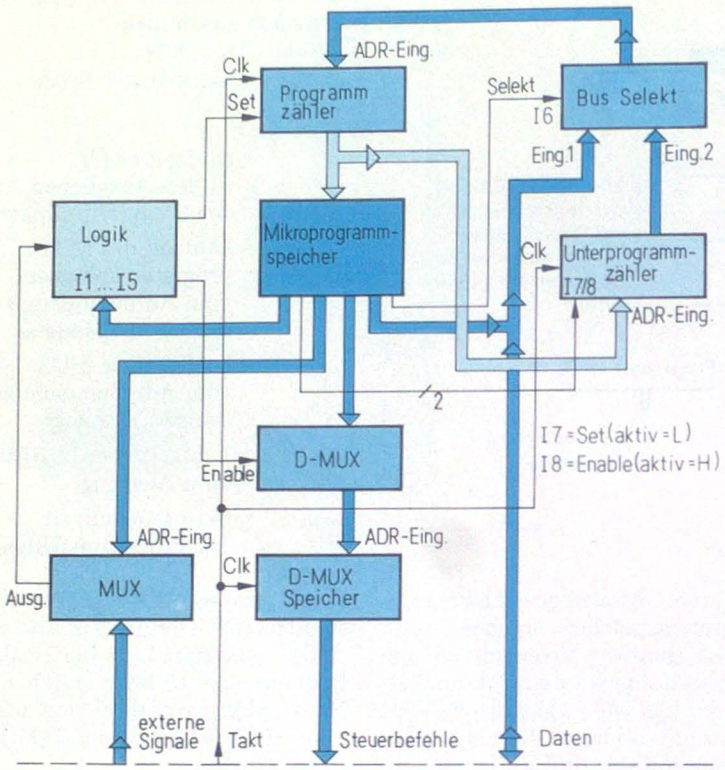


Bild 5. Mit einigen Zusätzen sind auch Unterprogramme möglich, um eine effektivere Ausnutzung des Programmspeichers zu ermöglichen

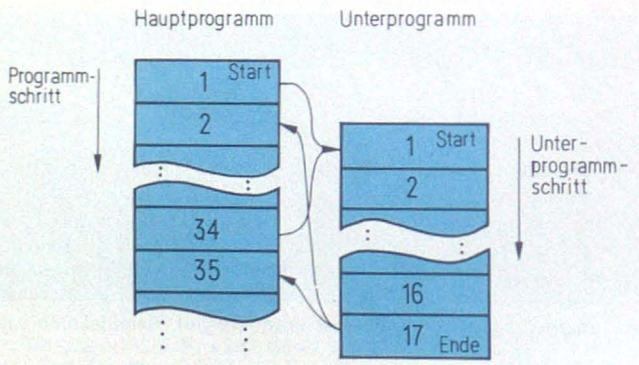


Bild 6. Unterprogramme können beliebig oft vom Hauptprogramm aufgerufen werden und kehren jeweils zum nächsten Programmschritt ins Hauptprogramm zurück

Tabelle 4: Organisation der Eingangs- und Ausgangsbefehle im Beispiel

Eingangsbit	MUX-Adresse Dez.	BCD
Person will kommen/gehen	1	LLH
Tür ist geöffnet	2	LHL
Person befindet sich außen im Vorhof	3	LHH
Person befindet sich innen im Raum	4	HLL
Beleuchtung ist innen notw.	5	HLH
Beleuchtung ist außen notw.	6	HHL

Wenn der Steuerbefehl ausgeführt werden soll, so soll das Steuerbit L-Pegel besitzen.

Steuerbit	D-MUX-Adresse Dez.	BCD
Keine Operation	0	LLL
Türe öffnen	1	LLH
Türe schließen	2	LHL
Beleuchtg. innen einschalten	3	LHH
Beleuchtg. innen ausschalten	4	HLL
Beleuchtg. außen einschalten	5	HLH
Beleuchtg. außen ausschalten	6	HHL

Ausbaumöglichkeiten

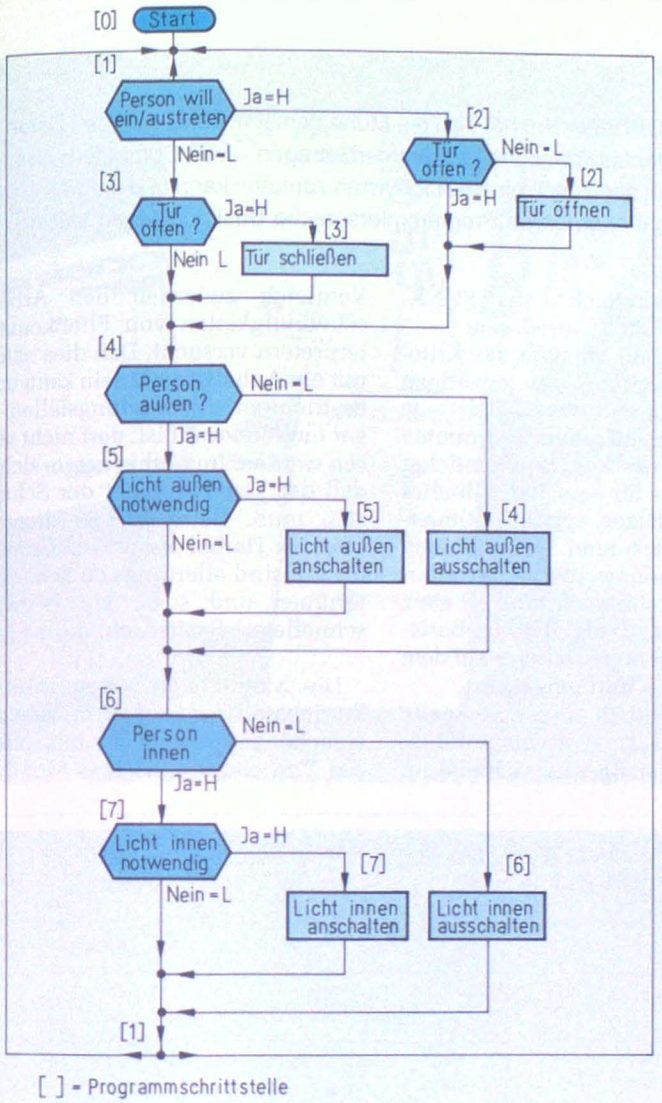
Die Steuereinheit kann als CCU in Verbindung mit einer arithmetischen Recheneinheit (ALU) zu einem Mikrocomputer zusammengeschaltet werden. Die Intelligenz dieses Computers hängt von der Leistunsfähigkeit der CCU und ALU ab. Die CCU wirkt in der Zusammenschaltung wie eine gewöhnliche Steuereinheit; die ALU ist der zu steuernde Teil. *Bild 8* zeigt den prinzipiellen Aufbau eines Mikroprozessors.

Ein weiterer Schritt in der beschriebenen Mikroprogrammierung führt zu der Bit-Slice-Technik. Für den Leser, der sich noch ausführlicher über die Mikroprogrammierung informieren möchte, sei deshalb Literatur in dieser Richtung empfohlen.

Tabelle 5: Mikroprogramm für unser Beispiel

Progr.-Schritt	Mikrobefehl: Sprungadresse Dez.	BCD	D-MUX-Adresse Dez.	BCD	MUX-Adresse Dez.	BCD	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	Kommentar
0	0	LLL	0	LLL	0	LLL	H	Z	Z	L	L	L	H	L	Ruhestellung
1	3	HLL	0	LLL	1	LLH	H	Z	H	H	L	L	H	L	IMPNC -Operation
2	4	HLL	1	LLH	2	LHL	L	H	Z	Z	H	L	H	L	IMP und OUTNC -Operation
3	Z	ZZZ	2	LHL	2	LHL	L	H	Z	L	L	L	H	L	OUTC -Operation
4	6	HHL	6	HHL	3	LHH	L	H	H	H	L	L	H	L	IMPNC und OUTNC -Operation
5	Z	ZZZ	5	HLH	6	HHL	L	Z	L	L	L	H	L		OUTC -Operation
6	1	LLH	4	HLL	4	HLL	H	H	H	L	L	H	L		IMPNC und OUTNC -Operation
7	1	LLH	3	LHH	5	HLH	L	Z	Z	H	L	H	L		IMP und OUTC -Operation

Der Mikrobefehl setzt sich aus I_1...I_8 und der BCD-codierten Adressen zusammen. Die Dezimalangabe dient nur zur besseren Übersicht.



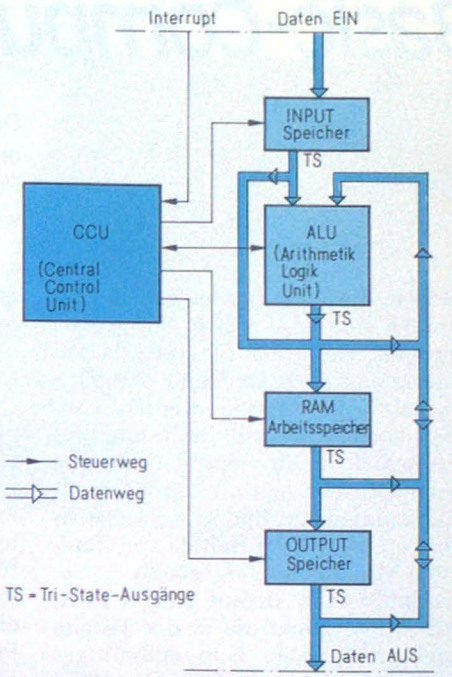


Bild 8. Ein Mikroprozessor unterscheidet sich intern nur wenig von einer Mikroprogramm-Steuerung. Jeder Befehl wird als Folge von Mikroprogramm-Befehlen verarbeitet

◄ Bild 7. Mit einer einfachen Mikroprogramm-Steuerung läßt sich zum Beispiel eine Tür automatisch öffnen und schließen und die Beleuchtung steuern. Hier das Flußdiagramm des Programmablaufes

Literatur

[1] *Klein, R.-D.*: Mikroprogramm-Steuerung. FUNKSCHAU 1978, Heft 23.

[2] *Nemec, J.; Sim, Gordon; Willis, B.*: A primer on bit slice processors: Electronic Design Vol. 25 (1977) Seite 52...60.

[3] Bit slice application Am 2900-System, Advanced Micro Devices.

[4] *Shavit, M.*: An emulation of the Am 9080 A: Application, Advanced Micro Devices.

[5] *Mick, J. R.; Brick, J.*: Microprogramming handbook: Application, Advanced Micro Devices.

[6] Build a microcomputer, Chapter I, II, III: Advanced Micro Devices.

[7] *Timm, V.*: TTL-Programmschaltwerk mit PROM; Elektronik, Bd. 24 (1975) Seite 53...58.

[8] *Feulner, W.*: Getaktete Logik mit Festwertspeichern; Elektronik, Bd. 26 (1977) Seite 99...100, Seite 133...134.

Stichworte zum Inhalt
Mikroprogrammsteuerung, Bit-Slice-Prozessor, Mikrobefehle, bedingte Sprünge, Unterprogramme, Taktfrequenz

Schreib-Lese-Test beim Nascom

Das im *Bild* als Hex-Dump abgedruckte Programm läuft auf dem Mikrocomputer Nascom-1 mit den Monitorprogrammen Nasbug-2, Nasbug-4 und Nassyn. Nach dem Start bei 0C50 löscht es den gesamten RAM-Speicher ab der Adresse 0C8B, d.h. es belegt ihn mit Null-Bytes. Dabei wird auch die Schreib-Lese-Funktion von Speichern und CPU überprüft. Nach dem Durchlauf erscheint die höchste getestete Adresse auf dem Bildschirm, und die Peripherial-Drive-LED blinkt.

Das Programm läuft mit geringen Änderungen auch auf anderen Z-80-, 8080- oder 8085-Systemen, ist aber nicht „relocatable", d.h. ohne Korrekturen verschieblich. *Manfred Segelke*

```
0c50  dd 21 8b 0c 3e aa dd 36
0c58  00 aa dd be 00 fa 69 0c
0c60  dd 36 00 00 dd 23 c3 56
0c68  0c dd 22 8b 0c 3a 8c 0c
0c70 cd 44 02 3a 8b 0c  cd 44
0c78  02 0e ff 79 08 af 3d 20
0c80  fd 08 3d 20 f7 cd 51 00
0c88  c3 7b 0c 00 00 00 00 00
```

Dieses kurze Programm löscht den RAM-Bereich ab 0C8B und überprüft auch die Schreib-Lese-Funktion

Basic-Computer

Die FUNKSCHAU-Redaktion hat sich die Mühe gemacht, alle wichtigen Daten heute erhältlicher „Personal Computer" zusammenzutragen – von Computern also, deren Preis man zur Not auch noch einem Hobbyisten zumuten kann und die sich normalerweise in der leicht erlernbaren Programmiersprache Basic bedienen lassen.

Unsere Basic-Computer-Tabelle beschränkt sich absichtlich auf solche Computer, die primär für das Arbeiten in Basic ausgelegt sind und praktisch einschaltfertig geliefert werden können. Nachträglich mit Basic-Interpretern ausrüstbare Einplatinen-Computer (Nascom-1, KIM-1 usw.) wurden nicht aufgenommen, sie finden sich aber in einer bereits 1979 in Heft 23 veröffentlichten Mikrocomputer-Tabelle.

Es muß auch darauf hingewiesen werden, daß allein die in der Tabelle aufgeführten Daten kein endgültiges Entscheidungskriterium für die Wahl eines bestimmten Geräts darstellen können. In diesem Zusammenhang sei auch auf den Beitrag „Computer im Small-Business-Bereich" aus FUNKSCHAU 1980, Heft 7, verwiesen.

Bei der Auswahl ist auch das Kriterium der Verbreitung des jeweiligen Computers nicht zu unterschätzen – je mehr Anwender mit einem bestimmten Gerät arbeiten, desto wahrscheinlicher ist es, daß man für sein individuelles Problem ein fertiges, system-optimiertes Programm bekommt. Dies gilt auch für Basic-Programme; die Syntaxunterschiede der Befehlsworte sind oft ganz erheblich, so daß ein TRS-80-Basic-Programm nicht ohne weiteres auf dem PET-2001 läuft – und umgekehrt.

Die Tabelle enthält auch eine Spalte „Geschwindigkeit". Hier wurde mit einer einfachen Verzögerungsschleife ein Vergleich zwischen den Arbeitsgeschwindigkeiten von Fließkomma-Interpretern versucht. Daß dies natürlich nur ein Anhaltspunkt sein kann und bei bestimmten Anwendungsfällen auch gar nicht relevant ist, darf nicht vergessen werden. Immerhin zeigte sich aber, daß der Teuerste nicht der Schnellste sein muß. Beim WH-89 diente das „Benton Harbor Basic" als Grundlage; für ihn sind allerdings auch andere Interpreter und sogar ein (wesentlich schnellerer) Basic-Compiler zu haben.

Die Möglichkeit, ohne zusätzliche Betriebssoftware sofort in Maschinensprache zu programmieren, erfordert das Vorhandensein eines Monitorpro-

Typ	PSI-80	HP-85	PC-1000	WH-89	Apple-II bzw. ITT-2020	AIM-65 bzw. PC-100
Hersteller	Kontron	Hewlett-Packard	Siemens	Heathkit	Apple Co. bzw. ITT	Rockwell bzw. Siemens
Mikroprozessor	Z 80 A	HP-interner Typ (8 bit)	6502	Z 80 A	6502	6502
Preis der Grundversion	10 000 DM	9000 DM	4000 DM	6000 DM	4000 DM	1400...2300 DM
RAM-Kapazität der Grundversion	48 KByte	16 KByte	32 KByte	16 KByte	16 KByte	1 oder 4 KByte
Basic-Interpreter	16 KByte (Floppy)	32 KByte ROM	24 KByte ROM	8 KByte (Floppy)	10 KByte ROM	8 KByte ROM
Maschinensprache-Monitor	ja	–	–	ja	ja	ja
Maximale interne RAM-Erweiterung	80 KByte	32 KByte	32 KByte	48 KByte	48 KByte	4 KByte
Geschwindigkeit (FOR I=0 TO5000: NEXT)	6 s	8,9 s	6 s	18 s	4,5 s	5 s
Ausgabe	Monitor S/W, 25 x 80 Zeichen; Grafik: 512 x 256 Punkte	Monitor S/W, Grafik: 256 x 192 Punkte; 16 x 32 Zeichen; Thermodrucker	Farb-Video-Ausg., Grafik	Monitor S/W, 25 x 80 Zeichen	Farb-Video-Ausgang, Grafik: 192 x 360 Punkte, 24 x 40 Zeichen	Thermodrucker und 20st. LED-Zeile (Drucker kann 100 Pkte. hor. plotten)
Eingebaute/mitgel. Massenspeicher	Floppy	Datenkassette	Floppy	Floppy-Option	–	–
Anschließbare Massenspeicher	Floppy	Floppy	Floppy	Kassetten-recorder	Kassetten-recorder, Floppy	Kassetten-recorder, Floppy
Schnittstellen	IEC-Bus; RS-232	RS-232; IEC-Bus	RS-232	RS-232	I/O-Port; RS-232-Option	TTY (20 mA); 2 I/O-Ports (VIA)

Der AIM 65 - hochbegabt für den Unterricht.

Eingebauter Drucker, leistungsfähige Zentraleinheit R6502, vielseitige Ein- und Ausgabemöglichkeiten, kurz: „summa cum laude".

Micro Power

Er ist der Primus seiner Klasse. Weil er speziell so konzipiert wurde, daß man die Arbeitsweise eines Mikrocomputers leicht im Unterricht lernen kann.

Rockwells AIM 65 – das vollständige Mikrocomputersystem für den Unterricht. Zu einem Preis, der für jeden Schuletat erschwinglich ist.

Einmalig in seiner Preisklasse: sein eingebauter thermischer Drucker. Er druckt Protokolle der Übungsaufgaben zur schnellen Nachprüfung durch Schüler und Lehrer. Die Schnittstellen ermöglichen den Anschluß z.B. von zwei Kassettenrecordern, Teletype und anderem mehr.

Bus- und System-Erweiterungen sind vorgesehen. Genauso wie der Ausbau mit PROMs, ROMs und RAMs.

Im Dialogverkehr über Monitor hilft der AIM 65 dem Schüler Schritt für Schritt beim Einstieg in die Mikroprozessor-Grundlagen. Und zwar inklusive Text-Editor, interaktivem mnemonischen Assembler, Fehlersuche (Trace, Break points) und vieles mehr.

Auf Wunsch lieferbar: ein symbolisches Assembler-Programm. Damit wird der AIM 65 ein leistungsstarkes Instrument für die Entwicklung von Mikrocomputern und die Herstellung von Prototypen.

Für den Fortgeschrittenen bietet ein zusätzlicher BASIC-Interpreter (resident in einem ROM) die Möglichkeit, auch höhere Computersprachen zu lernen. Ein Handbuch gibt weitere Hilfestellungen dazu.

Wenn Sie sich mit dem AIM 65 beschäftigen, werden Sie schnell entdecken, daß er sich auch hervorragend für die Steuerung und Überwachung von Laborgeräten eignet.

Mit geringem Aufwand können Sie mehrere AIM 65-Systeme miteinander kombinieren. Das erlaubt eine praxisnahe, effektive Schulung an und mit dem Mikroprozessor. Weil die Schüler nicht Schlange stehen müssen, gibt es keine Wartezeiten. Prüfen Sie den AIM 65 von Rockwell, seine überragende Ausstattung und den günstigen Preis.
Der AIM 65 wird die Prüfung mit „summa cum laude" bestehen.

Für weitere Informationen wenden Sie sich bitte an

System Kontakt GmbH,
Siemensstraße 5,
7107 Bad Friedrichshall,
Tel.: (0 71 36) 50 31,
Tx.: 728 563.

In Österreich an:
W. Moor Ges.m.b.H.
Storchengasse 1/1/1,
A-1150 Wien,
Tel.: (222) 85 86 46,
Tx.: 135701.

Rockwell International

...where science gets down to business

Typ	TRS-80	PET-2001	CBM-3016/ CBM-3032	Alphatronic	ABC-80	TI-99/4
Hersteller	Tandy	Commodore	Commodore	Triumph-Adler	Luxor	Texas Instruments
Mikroprozessor	Z80A	6502	6502	8085 A	Z80A	TMS 9900 (16 bit)
Preis der Grundversion	1700 DM	1800 DM	3000 DM	2250 DM	3200 DM	2400 DM
RAM-Kapazität der Grundversion	4 KByte	8 KByte	16 KByte bzw. 32 KByte	4 KByte	16 KByte	16 KByte
Basic-Interpreter	4 KByte (Level 1)	8 KByte ROM	8 KByte ROM	8 KByte ROM	16 KByte ROM	26 KByte ROM
Maschinensprache-Monitor	–	–	ja	ja	–	–
Maximale interne AM-Erweiterung	16 KByte	8 KByte	32 KByte	48 KByte	32 KByte	16 KByte
Geschwindigkeit (FOR I=0 TO5000· NEXT)	6,5 s (Level 2)	5,5 s	5,5 s	5,5 s	4,7 s	14,9 s
Ausgabe	Monitor S/W, 16 x 64 Zeichen	Monitor S/W, 25 x 40 Zeichen	Monitor S/W, 25 x 40 Zeichen	Videoausgang S/W 24 x 80 Zeichen	Monitor S/W, 24 x 40 Zeichen	Farb-Video-Ausg. Grafik: 192 x 256 Punkte; 24 x 32 Zeichen
Eingebaute/mitgel. Massenspeicher	Kassetten-recorder	Kassetten-recorder	–	Option: Floppy	Kassetten-recorder	–
Anschließbare Massenspeicher	Floppy	2. Recorder; Floppy	Kassettenrecorder, Floppy	Kassettenrecorder, Floppy	Floppy (bisher nicht lieferbar)	Kassettenrecorder, Floppy
Schnittstellen	RS-232-Option	IEC-Bus; I/O-Port (VIA)	IEC-Bus; I/O-Port (VIA)	RS-232	IEC-Bus-Option; RS-232	–

Typ	MZ-80K	Superboard (Challenger)	Proteus III C	DAI
Hersteller	Sharp	Ohio Scientific	Proteus/Corex	DAI (Brüssel)
Mikroprozessor	Z80	6502	6800	8080 A
Preis der Grundversion	2150 DM	890 DM	9400 DM	2500 DM
RAM-Kapazität der Grundversion	20 KByte	4 KByte	32 KByte	12 KByte
Basic-Interpreter	14 KByte auf Kassette	8 KByte ROM	9 KByte auf Floppy	24 KByte ROM
Maschinensprache-Monitor	–	–	ja	ja
Maximale interne AM-Erweiterung	20 KByte	8 KByte	56 KByte	48 KByte
Geschwindigkeit (FOR I=0 TO5000: NEXT)	4 s	6 s	20 s	6 s
Ausgabe	Monitor S/W, 25 x 40 Zeichen Grafik: 50 x 80 Punkte	Video-Ausg. S/W, 24 x 24 Zeichen	Video-Option (640 DM), sonst nur über V-24-Schnittstelle	Farb-Video-Ausg., Grafik: 260 x 352 Punkte; 24 x 60 Zeichen
Eingebaute/mitgel. Massenspeicher	Kassetten-recorder	–	Dual-Floppy	–
Anschließbare Massenspeicher	–	Kassetten-recorder	Floppy	Kassettenrecorder, Floppy
Schnittstellen	Drucker-Interface Opt. (8 bit parall.)	–	RS-232/V-24 I/O-Port, IEC-Bus	RS-232

Adressen der Hersteller

Kontron Elektronik GmbH, Breslauer Str. 2, 8057 Eching
Hewlett-Packard, Berner Str. 117, 6000 Frankfurt 56
Siemens AG, Postfach 103, 8000 München 1
Heathkit Bausatz-Elektronik GmbH, Robert-Bosch-Str. 32–38, 6072 Dreieich-Sprendlingen
ITT/SEL AG, Östliche 132, 7530 Pforzheim
Rockwell International GmbH, Fraunhoferstr. 11a, 8033 München-Martinsried
Tandy Corp., Immermannstr. 57, 4000 Düsseldorf 1
Commodore GmbH, Frankfurter Str. 171–175, 6078 Neu-Isenburg
Triumph-Adler GmbH, Fürther Str. 212, 8500 Nürnberg
Luxor Electronic, Kieler Str. 555, 2000 Hamburg 54
Texas Instruments Deutschland, Haggertystr. 1, 8050 Freising
Sharp Electronics GmbH, Sonninstr. 3, 2000 Hamburg 1
Ohio-Scientific: Vertrieb durch Balü, Burchardplatz 1, 2000 Hamburg 1
Corex, Feldstr. 25, 6380 Bad Homburg 1
DAI, Data Applications International, Drève des Renards 6 Bte 8, 1180 Brussels, Belgien

Die angeführten Firmen liefern meist nicht direkt an Endverbraucher, können jedoch stets den nächstgelegenen Distributor nennen.

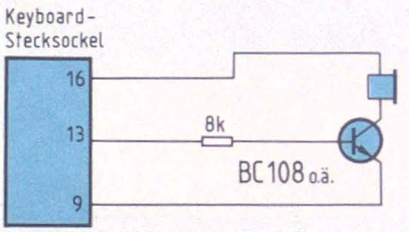
gramms, aber auch eine Benutzer-Unterstützung durch ausreichende Unterlagen über Adressenaufteilung und ROM-Unterprogramme. Schlechte Beispiele der oberen Preisklasse sind HP-85 und PC-1000 – hier beseitigte man jede Möglichkeit, in Maschinensprache zu programmieren –, und das beste Beispiel der unteren Preisklasse ist der AIM-65 bzw. PC-100, der eigentlich als kleines Entwicklungssystem brauchbar ist. Welchen Computer soll ich kaufen? Diese Frage kann unsere Tabelle sicher nicht endgültig beantworten. Vor Enttäuschungen kann man sich nur bewahren, wenn man sich vor dem Kauf genaustens überlegt, was man mit dem Computer eigentlich machen will. Für eine komfortable Textverarbeitung wird man einen Rechner mit 80 Zeichen pro Zeile und Floppy-Disk bevorzugen, während für Steueraufgaben oder für arithmetische Probleme auch ein Rechner mit alphanumerischem Display oder kleinerem Bildschirm ausreicht. Fe.

Z-80-Sirenenprogramm

F5 C5	Push AF, BC
3E 04	LD A, 04 (Start)
D3 00	Out (0), A
41	LD B, C
00	Nop
05	Dec B
20 FC	JR NZ
AF	XOR A
D3 00	Out (0), A
41	LD B, C
00	Nop
05	Dec B
20 FC	JR NZ
OC	Inc C
CD 69 00	Call KBD
30 E9	JR NC
C1 F1	Pop BC, AF
C9	Ret

Töne aus dem Nascom-1

Bei längeren und zeitaufwendigen Programmen können unnötige Wartezeiten vor dem Bildschirm vermieden bzw. anderweitig ausgenutzt werden, wenn sich der Computer am Programmende akustisch meldet. Von Interesse ist diese Möglichkeit auch bei interaktiven Programmen, bei denen der Mikrocomputer den Bedarf an Daten akustisch ankündigen kann. Beim Nascom ist dies zu realisieren, ohne daß einer der beiden Daten-Ports auf der Zentralplatine verwendet wird. Entsprechend dem *Bild* wird über eine Treiberstufe eine dynamische Hörkapsel an Pin 13 des Stecksockels auf der Tastatur-Platine angeschlossen. Dieser Datenausgang des Port O wird zum Abfragen des Tastenfeldes nicht benutzt. Er repräsentiert das Datenbit D_2. Das gleiche gilt übrigens für Pin 14, der das Datenbit D_5 führt. Die Spannungsversorgung kann ebenfalls am Stecksockel abgegriffen werden. Eine Überlastung des Port-ICs auf der Zentralplatine ist nicht zu befürchten, solange der Basiswiderstand ≧8 kΩ gewählt wird. Die ganze Zusatzschaltung kann bequem am Keyboard befestigt werden.



Anschluß einer dynamischen Hörkapsel an den Nascom. Statt der Kapsel kann auch die Serienschaltung eines kleinen 8-Ω-Lautsprechers und eines 100-Ω-Widerstandes verwendet werden

Sirenenprogramme gibt es bestimmt genug, trotzdem soll hier noch ein Listing gedruckt werden. Dieses kurze Programm hat den Vorteil, daß es von übergeordneten Programmen aufgerufen werden kann und auf die Betätigung einer beliebigen Taste des Keyboards die Kontrolle wieder an das übergeordnete Programm abgibt. Die Plazierung im Speicherbereich ist ebenfalls gleichgültig, da nur relative Sprünge verwendet werden.

Bernhard Wiedemann

Computer-Preise: Geht das mit rechten Dingen zu?

Wer regelmäßig einen Blick in die Anzeigenseiten der FUNKSCHAU wirft, wird schon festgestellt haben, daß für ein und denselben Computer oder auch programmierbaren Taschenrechner recht unterschiedliche Preise verlangt werden. Es muß nicht immer am preisgünstigsten sein, ein Gerät direkt von der Herstellervertretung zu erwerben. Ein Beispiel: Der HP-85 kostet in den USA 3250 Dollar (knapp 6000 DM), die deutsche HP-Vertretung „empfiehlt" einen Preis von 8920 DM. Blättert man nun die FUNKSCHAU durch, so findet man unschwer einen Anbieter, von dem man diesen Basic-Computer um fast 1000 DM billiger bekommt!

In einem kürzlich vom Südwestfunk gesendeten Interview schilderte Theo Liewen von der Firma Vero Computer (Aachen), was einem „Billiganbieter" widerfahren kann: Vero hatte vor einiger Zeit von dem Zwischenhändler Fritz Lehr Computer vom Typ PET-2001 gekauft und billiger als vom Hersteller Commodore empfohlen verkauft. Resultat: Lehr wurde nicht mehr mit PETs beliefert, und Vero mußte seine Geräte anderswoher beziehen. Mittlerweile hat jedoch Commodore den „Preisbrecher" verklagt, nicht mehr das Attribut „qualifizierter Fachhändler" verwenden zu dürfen – trotz bestem Service, trotz Selbstdiagnose-Programmpaket.

Selbstverständlich haben aber die Computerhersteller Argumente für die im Vergleich zu den USA überhöhten Preise parat: Hewlett-Pakkard verweist auf die hohen Kosten für die Erstellung deutscher Unterlagen, für das Training des Service-Personals und für die technische Umrüstung der Netzgeräte auf 220 V/50 Hz; Commodore ist gar der Ansicht, daß es nicht möglich sei, bei Unterschreitung empfohlener Endpreise noch Gewinn zu machen. Nur, das ist doch das Problem der Händler, nicht das der Hersteller – oder?! Fe

Rolf Kapust

TI-59 transformiert h-Parameter

Die h-Parameter, die man häufig in Transistor-Datenbüchern für bestimmte Grundschaltungen findet, erlauben eine präzise rechnerische Dimensionierung von Verstärkerstufen. Das hier vorgestellte TI-59-Programm gestattet das Transformieren dieser Parameter für andere Grundschaltungen.

Transistor-Grundschaltungen

Transistoren können in drei Grundschaltungen betrieben werden (*Bild 1*). Man spricht von Emitter-, Basis- oder Kollektorschaltung, je nachdem, welche der drei Elektroden gemeinsamer Pol für den Eingangs- und Ausgangskreis ist.

Der so entstehende Transistor-Vierpol läßt sich durch h-Parameter beschreiben (*Bild 2*), die aus Datenblättern entnehmbar oder aus den Kennlinienfeldern ablesbar sind. Die h-Parameter gelten nur für einen festen Arbeitspunkt. Die Bedeutung der h-Parameter ist in Bild 2 dargestellt.

Die Umrechnung der jeweiligen h-Parameter in solche einer anderen Grundschaltung geschieht im Programm über einen einfachen Transformationsalgorithmus. Zunächst wird aus den gegebenen h-Parametern eine Übergangsmatrix *h'* gebildet (*Bild 3a*). Je nachdem, in welche Grundschaltung umgerechnet werden soll, wird nun aus den Koeffizienten dieser Übergangsmatrix *h'* eine neue Matrix *h''* erzeugt (*Bild 3b*). Die gesuchte *h*-Matrix erhält man durch die gleiche Transformation wie eingangs (*Bild 3c*).

Mit den so umgerechneten h-Parametern können nach Eingabe eines Last- bzw. Generatorwiderstandes die interessierenden Größen des Kleinsignalverhaltens berechnet werden (*Bild 4*).

Programminstruktionen

Das Programm ist nach der Programmauflistung (*Bild 5*) einzugeben und zweckmäßigerweise auf Magnetkarte aufzuzeichnen. Es belegt die ersten 2 Blöcke des Programmspeichers des TI 59 und findet deshalb auf nur einer Magnetkarte Platz.

Die Handhabung des Programms ist in *Bild 6* als Kurzerläuterung wiedergegeben. Nach Betätigen der Taste A fordert das Programm die einzugebenden

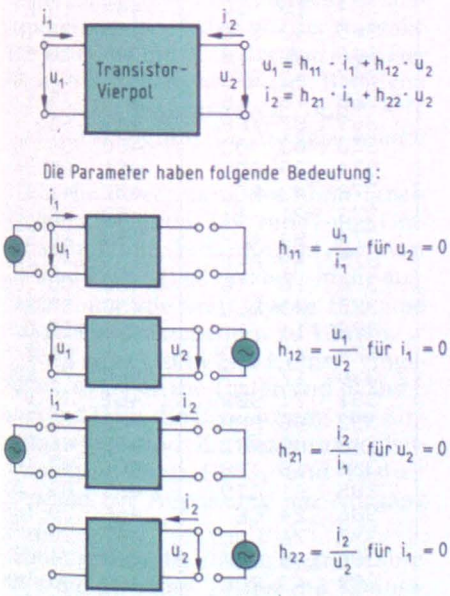


Bild 2. Beschreibung des Transistors durch h-Parameter. Sind die h-Parameter eines Transistors bekannt, so lassen sich für die jeweilige Grundschaltung die Kleinsignalgrößen berechnen

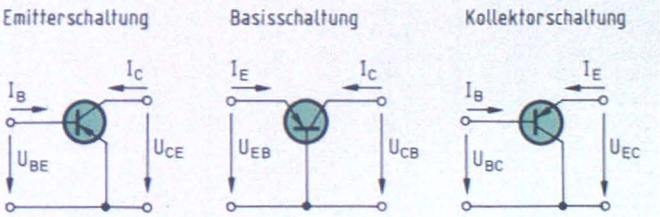


Bild 1. Grundschaltungen von Transistoren. Man spricht von Emitter-, Basis- oder Kollektorschaltung, je nachdem, welche der drei Elektroden der gemeinsame Pol für den Eingangs- und Ausgangskreis ist

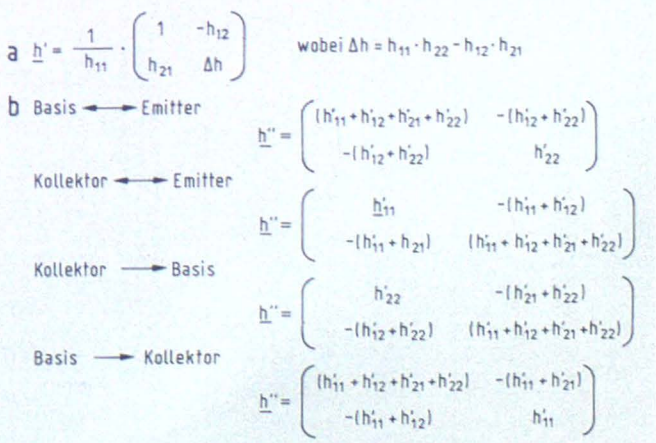
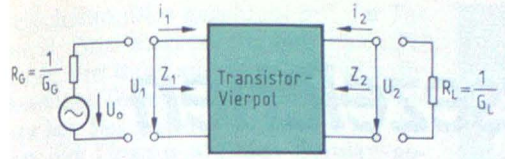


Bild 3. ▶
h-Parameter-Transformation, eine elegantere Umrechnungsmöglichkeit als mühsames „Zahlenschaufeln"

```
000  76 LBL
001  11  A
002  22 INV
003  52 EE
004  22 INV
005  57 ENG
006  87 IFF
007  00  00
008  42 STO
009  86 STF
010  00  00
011  01  1
012  01  1
013  42 STO
014  00  00
015  91 R/S
016  76 LBL
017  42 STO
018  72 ST*
019  00  00
020  71 SBR
021  29 CP
022  67  EQ
023  91 R/S
024  02  2
025  01  1
026  42 STO
027  00  00
028  76 LBL
029  91 R/S
030  43 RCL
031  00  00
032  91 R/S
033  11  A
034  76 LBL
035  16 A'
036  87 IFF
037  01  01
038  43 RCL
039  86 STF
040  01  01
041  01  1
042  01  1
043  42 STO
044  00  00
045  76 LBL
046  43 RCL
047  43 RCL
048  00  00
049  22 INV
050  57 ENG
051  66 PAU
052  57 ENG
053  73 RC*
054  00  00
055  42 STO
056  01  01
057  71 SBR
058  29 CP
059  67  EQ
060  44 SUM
061  02  2
062  01  1
063  42 STO
064  00  00
065  76 LBL
066  44 SUM
067  43 RCL
068  01  01
069  91 R/S
070  16 A'
071  76 LBL
072  29 CP
073  69 DP
074  20  20
075  43 RCL
076  00  00
077  55  ÷
078  02  2
079  95  =
080  22 INV
081  59 INT
082  29 CP
083  92 RTN
084  76 LBL
085  95  =
086  43 RCL
087  13  13
088  85  +
089  43 RCL
090  14  14
091  85  +
092  43 RCL
093  23  23
094  85  +
095  43 RCL
096  24  24
097  95  =
098  92 RTN
099  76 LBL
100  85  +
101  43 RCL
102  14  14
103  85  +
104  43 RCL
105  24  24
106  95  =
107  94 +/-
108  92 RTN
109  76 LBL
110  75  -
111  43 RCL
112  23  23
113  85  +
114  43 RCL
115  24  24
116  95  =
117  94 +/-
118  92 RTN
119  76 LBL
120  65  ×
121  43 RCL
122  13  13
123  85  +
124  43 RCL
125  14  14
126  95  =
127  94 +/-
128  92 RTN
129  76 LBL
130  55  ÷
131  43 RCL
132  13  13
133  85  +
134  43 RCL
135  23  23
136  95  =
137  94 +/-
138  92 RTN
139  76 LBL
140  35 1/X
141  43 RCL
142  11  11
143  35 1/X
144  42 STO
145  13  13
146  65  ×
147  43 RCL
148  12  12
149  65  ×
150  43 RCL
151  21  21
152  94 +/-
153  85  +
154  43 RCL
155  22  22
156  95  =
157  42 STO
158  24  24
159  43 RCL
160  12  12
161  94 +/-
162  55  ÷
163  43 RCL
164  11  11
165  95  =
166  42 STO
167  14  14
168  43 RCL
169  21  21
170  55  ÷
171  43 RCL
172  11  11
173  95  =
174  42 STO
175  23  23
176  92 RTN
177  76 LBL
178  69 DP
179  00  0
180  63 EX*
181  02  02
182  72 ST*
183  01  01
184  69 DP
185  21  21
186  69 DP
187  22  22
188  97 DSZ
189  00  00
190  69 DP
191  92 RTN
192  76 LBL
193  12  B
194  71 SBR
195  35 1/X
196  71 SBR
197  95  =
198  42 STO
199  11  11
200  71 SBR
201  85  +
202  42 STO
203  12  12
204  71 SBR
205  75  -
206  42 STO
207  21  21
208  43 RCL
209  24  24
210  42 STO
211  22  22
212  71 SBR
213  35 1/X
214  61 GTO
215  22 INV
216  76 LBL
217  13  C
218  71 SBR
219  35 1/X
220  43 RCL
221  13  13
222  42 STO
223  11  11
224  71 SBR
225  65  ×
226  42 STO
227  12  12
228  71 SBR
229  55  ÷
230  42 STO
231  21  21
232  71 SBR
233  95  =
234  42 STO
235  22  22
236  71 SBR
237  35 1/X
238  61 GTO
239  22 INV
240  76 LBL
241  14  D
242  71 SBR
243  35 1/X
244  43 RCL
245  24  24
246  42 STO
247  11  11
248  71 SBR
249  75  -
250  42 STO
251  12  12
252  71 SBR
253  85  +
254  42 STO
255  21  21
256  71 SBR
257  95  =
258  42 STO
259  22  22
260  71 SBR
261  35 1/X
262  61 GTO
263  22 INV
264  76 LBL
265  15  E
266  71 SBR
267  35 1/X
268  71 SBR
269  95  =
270  42 STO
271  11  11
272  71 SBR
273  55  ÷
274  42 STO
275  12  12
276  71 SBR
277  65  ×
278  42 STO
279  21  21
280  43 RCL
281  13  13
282  42 STO
283  22  22
284  71 SBR
285  35 1/X
286  76 LBL
287  22 INV
288  02  2
289  42 STO
290  00  00
291  01  1
292  01  1
293  42 STO
294  01  01
295  01  1
296  03  3
297  42 STO
298  02  02
299  71 SBR
300  69 DP
301  02  2
302  42 STO
303  00  00
304  02  2
305  01  1
306  42 STO
307  01  01
308  02  2
309  03  3
310  42 STO
311  02  02
312  71 SBR
313  69 DP
314  22 INV
315  86 STF
316  01  01
317  22 INV
318  57 ENG
319  25 CLR
320  91 R/S
321  76 LBL
322  17 B'
323  42 STO
324  10  10
325  42 STO
326  16  16
327  42 STO
328  18  18
329  43 RCL
330  11  11
331  42 STO
332  15  15
333  87 IFF
334  03  03
335  38 SIN
336  71 SBR
337  66 PAU
338  49 PRD
339  16  16
340  01  1
341  42 STO
342  17  17
343  43 RCL
344  22  22
345  49 PRD
346  18  18
347  61 GTO
348  24 CE
349  76 LBL
350  38 SIN
351  71 SBR
352  66 PAU
353  42 STO
354  17  17
355  43 RCL
356  22  22
357  49 PRD
358  18  18
359  61 GTO
360  24 CE
361  76 LBL
362  18 C'
363  42 STO
364  10  10
365  42 STO
366  18  18
367  42 STO
368  16  16
369  87 IFF
370  03  03
371  39 COS
372  43 RCL
373  21  21
374  94 +/-
375  49 PRD
376  16  16
377  43 RCL
378  11  11
379  42 STO
380  17  17
381  71 SBR
382  66 PAU
383  49 PRD
384  18  18
385  61 GTO
386  24 CE
387  76 LBL
388  39 COS
389  43 RCL
390  21  21
391  42 STO
392  16  16
393  01  1
394  42 STO
395  17  17
396  43 RCL
397  22  22
398  49 PRD
399  18  18
400  76 LBL
401  24 CE
402  43 RCL
403  15  15
404  85  +
405  43 RCL
406  16  16
407  95  =
408  55  ÷
409  53  (
410  43 RCL
411  17  17
412  85  +
413  43 RCL
414  18  18
415  54  )
416  95  =
417  91 R/S
418  76 LBL
419  66 PAU
420  43 RCL
421  11  11
422  65  ×
423  43 RCL
424  22  22
425  75  -
426  43 RCL
427  12  12
428  65  ×
429  43 RCL
430  21  21
431  95  =
432  92 RTN
433  76 LBL
434  34 FX
435  43 RCL
436  11  11
437  55  ÷
438  43 RCL
439  22  22
440  95  =
441  34 FX
442  92 RTN
443  76 LBL
444  10 E'
445  22 INV
446  86 STF
447  03  03
448  35 1/X
449  65  ×
450  06  6
451  95  =
452  42 STO
453  00  00
454  86 STF
455  40 IND
456  00  00
457  57 ENG
458  25 CLR
459  91 R/S
460  76 LBL
461  19 D'
462  71 SBR
463  66 PAU
464  87 IFF
465  03  03
466  30 TAN
467  35 1/X
468  76 LBL
469  30 TAN
470  34 FX
471  32 X:T
472  71 SBR
473  34 FX
474  65  ×
475  32 X:T
476  95  =
477  29 CP
478  91 R/S
479  00  0
```

Bild 5. Programmauflistung. Das h-Parameter-Programm füllt gerade die ersten beiden Blöcke des TI-59-Programmspeichers und läßt sich deshalb bequem auf nur einer Magnetkarte aufzeichnen



Eingangswiderstand $Z_1 = \frac{u_1}{i_1} = \frac{h_{11} + \Delta h \cdot R_L}{1 + h_{22} \cdot R_L}$

Ausgangswiderstand $Z_2 = \frac{u_2}{i_2} = \frac{h_{11} + R_G}{\Delta h + h_{22} \cdot R_G}$

Stromverstärkung $\frac{i_2}{i_1} = \frac{h_{21}}{1 + h_{22} \cdot R_L}$

Spannungsverstärkung $\frac{u_2}{u_1} = \frac{-h_{21} \cdot R_L}{h_{11} + \Delta h \cdot R_L}$

Anpassung am Ein- und Ausgang

Dabei sind:

$R_G = Z_1 = \sqrt{\frac{h_{11}}{h_{22}} \cdot \Delta h}$

$R_L = Z_2 = \sqrt{\frac{h_{11}}{h_{22}} \cdot \frac{1}{\Delta h}}$

Für die Determinante Δh gilt

$\Delta h = h_{11} \cdot h_{22} - h_{12} \cdot h_{21}$

Bild 4. Berechnungshinweise für die Kleinsignalgrößen bei Verwendung der h-Parameter

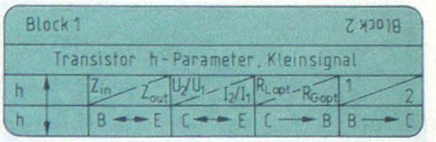


Bild 6. So läßt sich eine zweckmäßige Beschriftung der Magnetkarte am besten realisieren. Im Handel sind entsprechende Schreiber (z. B. LUMOCOLOR) in dünnen Strichstärken erhältlich

h-Parameter durch Angabe ihres Index an. Der so ausgewiesene h-Parameter wird über R/S eingegeben, wonach der nächste Index in der Anzeige erscheint. Eine Korrektur oder erneute Eingabe erfolgt nach Betätigen von RST und A. Danach beginnt die Eingabe wieder mit h_{11}.

Über die Tasten B, C, D oder E erfolgt die Umrechnung der h-Parameter von einer Grundschaltung in die andere (siehe Bild 6). Diese Umrechnung nimmt etwa 16 Sekunden Rechenzeit in Anspruch.

Die jetzt umgerechneten h-Parameter werden mit 2nd A abgerufen. Zur Kontrolle, um welchen h-Parameter es sich handelt, erscheint kurz vor der Ausgabe der jeweilige Index in der Anzeige. Der nächste Parameter wird nach Betätigen der R/S-Taste ausgegeben. Durch RST und 2nd A beginnt die Ausgabe wieder mit h_{11}.

Für die Berechnung der Kleinsignalgrößen übernimmt Lbl 2nd E eine Umschaltfunktion, da die Anzahl der „user defined keys" (Labeltasten) nicht ausreichte, um alle verfügbaren Ein- und Ausgaben direkt steuern zu können.

Wird zunächst in 2nd E eine 1 eingegeben, so gelten die Tasten 2nd B, 2nd C und 2nd D für die Berechnung von Eingangswiderstand Z_{in}, Spannungsübertragungsfunktion U_2/U_1 und Lastwiderstand bei Anpassung am Ausgang R_{Lopt}.

Mit der Eingabe einer 2 in 2nd E sind über die gleichen Tasten die Kleinsignalgrößen Ausgangswiderstand Z_{out}, Stromübertragungsfunktion I_2/I_1 sowie Generatorwiderstand bei Anpassung am Eingang R_{Genopt} berechenbar.

Für die Berechnung einiger Kleinsignalgrößen sind Last- oder Generatorwiderstand vorher einzugeben (*Tabelle*). Der einmal gewählte Ausgabemodus bleibt auch für weitere Berechnungen erhalten, soweit keine Änderung über 2nd E erfolgt. Erkennbar ist der Ausgabemodus durch technisches Anzeigeformat (ENG).

Eine zweckmäßige Beschriftung der Magnetkarte, wie sie in *Bild 6* wiedergegeben ist, erleichtert das Arbeiten mit dem Programm wesentlich.

Tabelle: Bedienung des Programms

Schritt	Verfahren	Eingabe	Taste	Anzeige
1	Eingabe der h-Parameter		A	11
	Der einzugebende Parameter wird	h_{11}	R/S	12
	durch Bekanntgabe seines Index	h_{12}	R/S	21
	in der Anzeige angefordert!	h_{21}	R/S	22
		h_{22}	R/S	⋮
	Korrektur oder Neueingabe		RST	
	beginnend mit h_{11}		A	11
		h_{11}	R/S	12
				⋮
2	Umrechnung von → in			
	wahlweise: Basis → Emitter		B	0
	Emitter → Basis		B	0
	Kollektor → Emitter		C	0
	Emitter → Kollektor		C	0
	Kollektor → Basis		D	0
	Basis → Kollektor		E	0
	eventuell weiter mit Schritt 4			
3	Ausgabe der umgerechneten Parameter		2nd A	„11" h_{11}
	Index des als nächstes ausgegebenen Parameters „erscheint" kurz vorher in der Anzeige!		R/S	„12" h_{12}
			R/S	„21" h_{21}
			R/S	„22" h_{22}
	Erneute Ausgabe beginnend mit h_{11}		RST 2nd A	
4	Vorwahl der auszugebenden Kleinsignalgrößen			
	für Z_{in}, U_2/U_1 oder R_{Lopt}	1	2nd E	0 00
	für Z_{out}, I_2/I_1 oder R_{Genopt}	2	2nd E	0 00
5	Je nach Vorwahl bei Schritt 4 Ausgabe geforderter Größen nach Eingabe von			
	R_L für Z_{in}	R_L	2nd B	Z_{in}
	R_{Gen} für Z_{out}	R_{Gen}	2nd B	Z_{out}
	R_L für U_2/U_1	R_L	2nd C	U_2/U_1
	R_L für I_2/I_1	R_L	2nd C	I_2/I_1
	keine Eingabe		2nd D	R_{Lopt}
			2nd D	R_{Genopt}

AIM-65 empfängt Morsezeichen

Nach der Veröffentlichung eines Programms, bei dem der AIM-65 bzw. PC-100 als Morseschreibmaschine fungiert, wollten zahlreiche Leser wissen, ob man mit diesem Mikrocomputer wohl Telegrafie auch empfangen könne. Natürlich kann man! Die dafür notwendigen theoretischen Grundlagen sind auf Seite 63 nachzulesen.

Das in *Bild 1* als AIM-65-Assembler-Listing abgedruckte Programm arbeitet nach den in diesem Heft dargelegten Kriterien für den Empfang von CW-Zeichen bei Geschwindigkeiten bis zu etwa 300 Buchstaben pro Minute. Es stützt sich auf das in Heft 16/1979 auf Seite 952 veröffentlichte Sendeprogramm, das Morsezeichen direkt als Tonfrequenz am Port PB 7 ausgibt und über einen die gesamte Page 3 umfassenden Bufferspeicher verfügt, damit man mit der AIM-Tastatur 255 Zeichen vorausschreiben kann.

Ein- und Ausgänge der Kombination beider Programme entsprechen den Angaben, die bei der Beschreibung eines Funkfernschreib-Interfaces in Heft 16/1979 auf Seite 963 gemacht wurden, und sind mit der Belegung bei den RTTY-Programmen in den Heften 15 und 16/1979 identisch. Beim PLL-IC NE 567 im Fernschreibinterface ist dabei von Pin 1 ein Widerstand von etwa 20...50 kΩ nach +5 V zu legen, solange man CW empfangen möchte. Andernfalls würde bereits das Empfänger-Hintergrundrauschen als Signal ausgewertet. Der 1-µF-Elko links von D3 ist im Interface bei Bedarf zu vergrößern, damit die PTT-Steuerung zwischen den einzelnen Morsezeichen nicht dauernd den Sender kurz ausschaltet.

Das Empfangsprogramm läßt sich manuell mit den Tasten 0...9 auf die gewünschte Geschwindigkeit einstellen, wobei Taste 0 der niedrigsten, Taste 9 der höchsten CW-Geschwindigkeit entspricht. Es genügt ein kurzes Antippen der entsprechenden Taste; der eingegebene Wert bleibt auch erhalten, wenn man auf das Sendeprogramm

```
0000          ;*********************
0000          ;*** CW DECODER ***
0000          ;*********************
0000 PA             =$A00F      ;INPUT=PA0
0000 SHIFT          =$40        ;DISPLAY
0000 TX             =$200       ;TRANSM.PGM
0000 CWT            =0          ;TABLE
0000 GET            =$ED05      ;SCAN KB
0000 ASC            =$EC85      ;KB/ASCII
0000 LEN            =$FF
0000 SP             =$FE        ;SPEED
0000 PAU            =$FD
0000 CHR            =$FC
0000 CNT            =$E8        ;TIMING
0000                *=$242
0242 CW     A201    LDX #1      ;START
0244        86FC    STX CHR
0246 ELEM   A200    LDX #0
0248        86FF    STX LEN
024A CW1    A5FE    LDA SP
024C        85E8    STA CNT
024E KEY    2005ED  JSR GET     ;TEST KB
0251        88      DEY         ;CORR KB ROW
0252        2085EC  JSR ASC     ;CALC.ASCII
0255        C953    CMP #'S'    ;EXIT
0257        F0A7    BEQ TX      ;IF S KEY
0259        C93A    CMP #$3A
025B        100A    BPL DLY
025D        E92F    SBC #$2F    ;'0'=$30
025F        3006    BMI DLY
0261        AA      TAX         ;LOOKUP
0262        BDB102  LDA CWSP,X  ;SPEED
0265        85FE    STA SP
0267 DLY    C6E8    DEC CNT
0269        D0E3    BNE KEY
026B        AD0FA0  LDA PA      ;TEST INP
026E        2901    AND #1      ;MASK PA0
0270        D008    BNE CW2     ;NO SIGNAL
0272        A9FF    LDA #$FF
0274        85FD    STA PAU
0276        E6FF    INC LEN
0278        D0D0    BNE CW1
027A CW2    E6FD    INC PAU
027C        A5FD    LDA PAU
027E        F0CA    BEQ CW1
0280        C908    CMP #8
0282        F00E    BEQ CW3     ;CHR.READY
0284        B023    BCS CW4
0286        A5FF    LDA LEN
0288        C903    CMP #3      ;NOISE?
028A        90BA    BCC ELEM    ;YES
028C        C907    CMP #7      ;DOT/DASH
028E        26FC    ROL CHR     ;INTO BUFFER
0290        D0B4    BNE ELEM    ;UNCOND.JMP
0292 CW3    26FC    ROL CHR     ;ADJ FOR
0294        90FC    BCC CW3     ;TX TABLE
0296        A5FC    LDA CHR
0298        A23F    LDX #$3F    ;CONVERT TO
029A CNV    D500    CMP CWT,X   ;ASCII
029C        F003    BEQ OFFS
029E        CA      DEX
029F        D0F9    BNE CNV
02A1 OFFS   8A      TXA         ;ADD $20
02A2        691F    ADC #$1F    ;TO ASCII
02A4 DIS    204000  JSR SHIFT   ;DISPL
02A7        3099    BMI CW      ;NEXT CHAR.
02A9 CW4    C90F    CMP #15
02AB        D095    BNE CW
02AD        A920    LDA #$20    ;SPACE
02AF        D0F3    BNE DIS
02B1 CWSP   2A      .BYT 42,33  ;SPEED
02B2        21
02B3        1B      .BYT 27,21,17,14
02B4        15
02B5        11
02B6        0E
02B7        0B      .BYT 11,9,7,5
02B8        09
02B9        07
02BA        05
```

Bild 1. AIM-Assemblerlisting des Programms zum Empfang von Telegrafie-Zeichen (CW). Wer's eilig hat, braucht nur mit dem „/"-Befehl die mittlere Spalte einzugeben (A2 01 86...)

umschaltet. Dies geschieht mit der Taste S. Anzumerken ist, daß das „Timing" erst dann stimmt, wenn man die jeweilige Taste (0...9) wieder losgelassen hat. Bei gedrückter Taste wird daher nur Unsinn auf das Display geschrieben. Die Zeitkonstanten stehen an den Adressen 02B1 bis 02BA.

Für alle, die Heft 16/1979 gerade nicht zur Hand haben, gibt *Bild 2* das Sendeprogramm mit den notwendigen Änderungen an. Jene Leser, die dieses Programm bereits auf Kassette besitzen, sollten die Änderungen an den Adressen 022C und 00B3 beachten. Die Taste F2 dient jetzt dazu, vom Sende- in das Empfangsprogramm umzuschalten, sobald das letzte Zeichen aus dem Bufferspeicher ausgegeben ist. Nach wie vor lassen sich die Betriebsparameter des Sendeprogramms individuellen Wünschen anpassen:

0062 CW-Geschwindigkeit;
0085 Pause zwischen Zeichen;
009C Pause zwischen Worten;
0215, 021A Tonfrequenz.

Eine Erprobung des Decoders ist auch ohne Kurzwellenempfänger möglich. Dabei ist eine Morsetaste zwischen dem Port PA 0 und Masse anzuschließen.

Je nachdem, ob man zuerst senden oder zuerst empfangen will, kann man das Gesamtprogramm entweder bei 0200 oder 0242 starten. Im praktischen Betrieb zeigt sich schnell, daß die Pausen zwischen den Zeichen oft am unsaubersten gesendet werden, so daß nicht immer richtig zwischen Zeichen- und Wortzwischenräumen unterschieden wird. An diesem Kriterium ist auch eine nicht ganz exakte Geschwindigkeits-Einstellung zu erkennen.

Die Code-Umwandlungstabelle (0000...003F) enthält im Listing von Bild 2 noch keine Sonderzeichen. Eine entsprechende Erweiterung ist aber leicht möglich. Die Adresse des Codes ergibt sich aus dem um hex 20 verminderten ASCII-Wert der entsprechenden AIM-Taste, und alle Bytes, die in Bild 2 nur „80" enthalten, stehen einer Sonderzeichen-Belegung offen.

Herwig Feichtinger

Quizfrage für Spezialisten

Zwei Bytes können in Bild 1 durch hex EA (NOP) ersetzt werden, ohne daß sich für den Benutzer irgend etwas ändert. Zu schwierig? Kleine Hilfe: Es handelt sich um einen Sprungbefehl. Alles klar? Die Auflösung folgt im nächsten Heft.

```
0000 80 56 80 80    0054 D0 EE A5 FB    00A8 A2 FF 9A A9
0004 80 80 80 80    0058 60 00 00 00    00AC 00 8D 0B A0
0008 80 80 80 80    005C 48 C6 E5 D0    00B0 8D 0E A0 4C
000C CE 8C 56 94    0060 42 A9 80 85    00B4 42 02 00 00
0010 FC 7C 3C 1C    0064 E5 C6 E4 F0    0200 A9 5C 8D 04
0014 0C 04 84 C4    0068 14 10 38 06    0204 A4 A9 00 8D
0018 E4 F4 56 56    006C E3 F0 15 B0    0208 05 A4 A9 40
001C 80 80 80 32    0070 04 A9 01 D0    020C 8D 0B A0 A9
0020 4C 60 88 A8    0074 02 A9 03 85    0210 C0 8D 0E A0
0024 90 40 28 D0    0078 E4 A9 C0 D0    0214 A9 00 8D 04
0028 08 20 78 B0    007C 02 A9 40 8D    0218 A0 A9 02 8D
002C 48 E0 A0 F0    0080 0B A0 D0 1F    021C 05 A0 A9 00
0030 68 D8 50 10    0084 A9 02 85 E4    0220 A2 16 95 E6
0034 C0 30 18 70    0088 86 E2 A6 E6    0224 CA 10 FB 20
0038 98 B8 C8 80    008C E4 E7 F0 11    0228 40 EC C9 0D
003C 80 00 80 80    0090 BD 00 03 F0    022C 58 EA EA EA
0040 85 FC A2 00    0094 15 85 E3 C9    0230 EA 20 40 00
0044 86 FF B5 E9    0098 80 D0 04 A9    0234 AA BD E0 FF
0048 95 E8 09 80    009C 08 85 E4 E6    0238 A6 E7 E6 E7
004C 20 7B EF A6    00A0 E6 A6 E2 2C    023C 9D 00 03 18
0050 FF E8 E0 14    00A4 04 A0 68 40    0240 90 E5 A2 01
```

Bild 2. Hier noch einmal das CW-Sendeprogramm mit den erforderlichen Änderungen gegenüber Heft 16/1979

Zwei TI-59-Nachträge

FUNKSCHAU 1980, Heft 5

RLC-Netzwerksberechnung

Meistens möchte man das Ergebnis nicht in Real- und Imaginärteil vorliegen haben, sondern als Scheinwiderstand und Phasenverschiebung. Mit Hilfe des P → R-Befehls gelingt dies leicht. Ein kurzes Programmstück mit P→R liefert den Scheinwiderstand und nach Drücken von R/S den Phasenwinkel.

Wenn man vergißt, vor Beginn der Rechnung die Speicher zu löschen, erhält man falsche Ergebnisse. Um das zu vermeiden, könnte man den Cms-Befehl als ersten Befehl nach Label D einfügen und die Sprungadressen entsprechend ändern.

Primfaktoren-Bestimmung

Dieses Programm wird durch zwei grundlegende Mängel in seiner Laufzeit unnötig verlängert. Erstens gibt es nach 2 keine geraden Primfaktoren, man kann also ab 3 die Laufvariable in Zweierschritten hochzählen und spart dadurch bereits fast die Hälfte der Laufzeit ein.

Die entscheidende Beschleunigung bringt aber folgende Überlegung: Es gibt zu einer Zahl keinen Primfaktor, der größer ist als die Wurzel dieser Zahl, wenn nicht gleichzeitig ein Primfaktor existiert, der kleiner ist als die Wurzel. Die Suche nach einem solchen kann also abgebrochen werden, wenn die Laufvariable größer als die Wurzel geworden ist, die Zahl selbst ist dann der Primfaktor. Die Einsparung an Laufzeit wird um so größer, je größer die Primfaktoren einer Zahl sind.

Ing. (grad.) Robert Lechner

Anm. d. Red.: Auf den Abdruck ausgeführter Programme verzichten wir hier, da die Implementierung der Rechenvorschriften in den TI-59 auch für den Anfänger kein Problem darstellen dürfte.

Zwei Mikroprozessoren in einem Computer

Der seit kurzer Zeit nun auch hierzulande erhältliche Computer WH-89 von Heathkit enthält gleich zwei Z-80-Mikroprozessoren: Einer dient nur für die Bildschirm-Organisation, mit dem anderen wird gerechnet. Der folgende Beitrag beschreibt interessante Details dieses Personal Computers der oberen Preisklasse.

Der Computer WH-89 gliedert sich intern in zwei Teile, die beide jeweils mit einem Z-80-Mikroprozessor ausgerüstet sind: ein intelligentes Terminal mit serieller Schnittstelle und dem eigentlichen Mikrocomputer, der die vom Benutzer eingegebenen Befehle ausführt.

Der Computer kann mit bis zu 48 KByte RAM-Speicher bestückt werden (*Tabelle 1*). Im ROM befindet sich ein relativ einfaches Monitorprogramm, das nur 2 KByte umfaßt und im wesentlichen nur dazu dient, das Betriebssystem von der ebenfalls schon eingebauten Floppy-Disk oder vom Kassettenrecorder zu laden. Diesen Vorgang bezeichnet man als „Bootstrapping". Interpreter für höhere Programmiersprachen und Assembler für die Programmierung in Maschinensprache müssen also grundsätzlich erst geladen werden. Dies macht den Umgang mit dem WH-89 gegenüber anderen Personal Computern besonders nach dem Einschalten deutlich komplizierter. Dem steht der Vorteil gegenüber, daß z. B. der Basic-Interpreter keinen Adressenraum belegt, wenn man nur Maschinensprache-Programme laufen läßt oder mit dem Assembler arbeitet.

Eingebaut ist auch ein Schwarzweiß-Bildschirm, auf dem sich 25 Zeilen mit je 80 Zeichen darstellen lassen. Jedes Zeichen wird dabei als Matrix von 5 x 7 Punkten generiert: Kleinbuchstaben wie g, p und q, die eine „Unterlänge" aufweisen, werden um zwei Punkte nach unten versetzt, so daß sich ein sauberes Schriftbild ergibt. Ferner kann man jedes Zeichenquadrat auch als Grafikelement mit 8 x 10 Punkten definieren. Eine Besonderheit ist auch, daß man vom Programm aus direkt auf die 25. Zeile des Schirms zugreifen kann. Alle Zeichen lassen sich ferner normal und invers (schwarz auf weiß) darstellen.

Zur Kommunikation mit der Außenwelt enthält der WH-89 zwei serielle RS-232-Schnittstellen für Baudraten bis 19 200 Bd. Das Kassetteninterface arbeitet nach der Kansas-City-Norm mit 1200 Bd, wobei jeder preiswerte Recorder geeignet ist, der sich ferngesteuert ein- und ausschalten läßt.

Wie schon erwähnt, ist der Basic-Interpreter hier nicht fest in ROMs gespeichert; zum Lieferumfang des WH-89 gehört daher unter anderem eine Diskette mit der erforderlichen Betriebs-Software. Derzeit existieren mehrere Basic-Versionen, zum Beispiel eine aus dem Hause Microsoft, woher auch die Interpreter für PET, Apple, TRS-80 und andere Personal Computer stammen. Ferner gibt es eine zu dem bereits in Heft 25/1979 ausführlich beschriebene H-8-System kompatible Basic-Version.

Der ebenfalls erhältliche Assembler krankt ein wenig daran, daß er einfach vom H-8-System übernommen wurde, so daß er zwar die 8080-Befehle, nicht aber die zusätzlichen des Z-80 beherrscht. Ein kleiner Mangel haftet auch der mitgelieferten Literatur an: Im Gegensatz zu dem kürzlich vorgestellten Lehrsystem ET-3400 des gleichen

Tabelle 1. Speicheraufteilung

0-	System-Monitor-ROM
800-	Reserviertes ROM
1000-	Reserviertes RAM
1400-	Floppy-Disk-RAM
1800-	Floppy-Disk-ROM
2000-	Anwender-RAM
E000-	Reserviert
FFFF	

Tabelle 2. Befehlssatz des Microsoft-Basic für den WH-89

ABS	AND	ASC	ATN
AUTO	CINT	CSNG	CDBL
CHR$	CLEAR	CLOSE	CONT
COS	DATA	DEF	DEFDBL
DEFINT	DEFSNG	DEFSTR	DEFUSR
DELETE	DIM	EDIT	ELSE
END	ERASE	ERL	ERR
ERROR	FIELD	FIX	FN
FOR	FRE	GET	GOSUB
GOTO	HEX$	IF	IMP
INP	INPUT	INSTR	INT
KILL	LEFT$	LEN	LET
LINE	LIST	LOAD	LIST
LSET	MERGE	MID$	MOD
NAME	NEW	NEXT	NOT
NULL	OCT$	ON	OPEN
OR	OUT	PEEK	POKE
POS	PRINT	PUT	READ
REM	RESET	RESTORE	RESUME
RETURN	RIGHT$	RND	RSET
RUN	SAVE	SGN	SIN
SPACE$	SQR	STOP	STR$
STRING$	SWAP	SYSTEM	TAN
THEN	TO	TROFF	TRON
USR	VAL	VARPTR	WAIT
WIDTH	XOR		

Der Computer WH-89 von Heathkit glänzt mit zwei eingebauten Z-80-Mikroprozessoren, von denen einer nur für Tastatur und Bildschirm da ist – mit dem anderen wird gerechnet

Herstellers läßt sie ein wenig an Systematik und vor allem Übersichtlichkeit vermissen. Auch ist es etwas schwierig, das System sofort nach dem Auspacken in Betrieb zu nehmen und z. B. ein Demonstrationsprogramm ablaufen zu lassen, weil man zunächst nicht recht weiß, welche Tasten man drücken muß, um ein Programm zu laden und zu starten.

Hat man aber einmal die Eigenschaften des WH-89 kennengelernt, so will man seine Vorzüge gegenüber anderen (meist allerdings billigeren) Personal Computern nicht mehr missen. Dazu gehört auch der recht komfortable Basic-Befehlssatz (*Tabelle 2*). Ihn könnte man etwa zwischen denen des PET-2001 und des Apple-II/ITT-2020 einordnen. Fe.

TI-59-Phantomcodes

Wer sich einmal intensiv mit den Tastencodes des TI-59 beschäftigt hat, dem ist sicherlich aufgefallen, daß von den 100 möglichen Codes (00...99) nur 92 verwendet werden. Die Funktion der verbleibenden 8 Codes soll im folgenden beschrieben werden. Als Vorbemerkung jedoch erst einmal eine Methode, die Phantomcodes in den Programmspeicher zu übernehmen.

Will man z. B. den Code 21 einprogrammieren, so tastet man STO 21 ein und geht mit BST BST auf den programmschritt STO zurück. Durch den Befehl 2nd DEL wird STO gelöscht, und im Programmspeicher steht nur noch 21 (vom Drucker übrigens als 2nd ausgedruckt).

Die 8 Phantomcodes und ihre vom Drucker angegebene Bedeutung lauten wie folgt:

21=2nd
Der Tastencode 21 zeigt während des Programmablaufs keine Wirkung. Steht er jedoch im Programmspeicher und wird durch SST zur Anwendung gebracht, so verhält er sich wie der manuell eingegebene Befehl 2nd.

26=2nd
Code 26 wird vom Drucker ebenfalls als 2nd identifiziert und zeigt keinerlei Wirkung.

31=LRN
Der Befehl 31 bewirkt im Programm ein automatisches Umschalten in den LRN-Modus.

41=SST
Dieser Befehl wird vom Rechner völlig übergangen. So bewirkt z. B. das Programm GTO SST 115 einen Sprung zu Schritt 115.

46=INS
Wirkt ähnlich wie 41, das Programmsegment GTO INS 115 wird jedoch als Sprung zum Label E (15) aufgefaßt, während die erste Ziffer 1 wie eine Zahleneingabe wirkt. Andererseits bewirkt GTO INS A E einen Sprung zu Label A.

51=BST
Gleiche Funktion wie 46.

56=DEL
Gleiche Funktion wie 46.

82=HIR

Dieser Befehl verdient sicherlich die größte Beachtung. Er verarbeitet den folgenden Befehl nicht als 0, sondern wie die Befehle STO, RCL usw. als 00. Wie Untersuchungen zeigten, ist er in der Tat etwas Ähnliches. Dieser Befehl erlaubt nämlich einen direkten Zugriff auf die internen Rechenregister des TI 59. Gibt man z. B. folgende Rechnung ein, belegt man sämtliche Rechenregister, was durch Blinken angezeigt wird, mit den Zahlen von 1 bis 9 (die 9 steht im Anzeigeregister):

1+2x(3+4x(5+6x(7+8x(9+

(nach dem letzten + blinkt die Anzeige). Die Befehle HIR 11 bis HIR 18 rufen nun nacheinander die Zahlen 1...8 aus den Registern zurück. Mit den Befehlen HIR 01 bis HIR 08 kann man beliebige Zahlen in den Rechenregistern 1 bis 8 unterbringen. Doch nicht genug des Guten. Man kann mit diesen Registern auch Speicherarithmetik betreiben. Die Befehle hierzu lauten:

STO ≙ HIR 01 bis HIR 08
RCL ≙ HIR 11 bis HIR 18
SUM ≙ HIR 31 bis HIR 38
PRD ≙ HIR 41 bis HIR 48
INV SUM ≙ HIR 51 bis HIR 58
INV PRD ≙ HIR 61 bis HIR 68 und
HIR 71 bis HIR 78 und
HIR 81 bis HIR 88 und
HIR 91 bis HIR 98

Die Bedeutung der Befehle HIR 09, 10, 19, 20, 21–30, 39, 40, 49, 50, 59, 60, 69, 70, 79, 80, 89, 90, 99 habe ich bisher noch nicht erkennen können.

Doch nach all diesen Vorteilen eine bittere Pille. Beim Gebrauch der Speicherarithmetik ist hier nämlich Vorsicht geboten, da der Rechner teilweise die eingegebenen Zahlen verändert. Wird eine Zahl aus dem Bereich 0.0000000001 bis 0.9999999999 oder –0.0000000001 bis –0.9999999999 eingegeben und mit Speicherarithmetik behandelt, ändert sie ihren Wert. Diese Änderung wird schon im Anzeigeregister durchgeführt. Aber auch trotz dieses Nachteils, der sich durch Verwendung des EE- oder Eng-Formats vermeiden läßt, ist der HIR-Befehl eine weitere Steigerung der Leistungsfähigkeit des ohnehin sehr leistungsstarken Zugpferdes von TI. *Frank Pradel*

Änderungen der Telegrafie-Codetabelle

Unser Leser C. Möller (DL7CM) schlägt für die AIM-65-Telegrafie-Programme (siehe auch FUNKSCHAU 1979, Heft 16) folgende Belegung der Sonderzeichen vor:

Adresse	ASCII	Telegrafie-Bedeutung	Hex-Code
0001	!	Unterstreichung	36
0002	"	Anführungszeichen	4A
0003	#	Irrung (hier 7 P.)	01
0004	$	Verkehrsende	16
0006	&	Bitte warten	44
0007	'	Apostroph	7A
0008	(	Klammer auf	B6
0009	)	Klammer zu	B6
000D	–	Bindestrich	86
001C	<	Spruchanfang	AC
001D	=	Trennung	8C
001E	>	Spruchende	54

Das Irrungszeichen sollte eigentlich acht Punkte umfassen; dies ist jedoch bei der 8-bit-Codierung von Morsezeichen leider nicht möglich, so daß nur sieben Punkte ausgestrahlt werden.

Henning Bülow

KIM als Sequencer

Ein immer häufiger in der Pop-, Rock- und besonders der elektronischen Synthesizer-Musik benutztes Gerät ist der Sequencer. Er erzeugt eine sich wiederholende Serie von mehreren Tönen, deren Höhe und Länge veränderbar sind. Auch der Mikrocomputer KIM-1 läßt sich in Verbindung mit einem Verstärker als Sequencer einsetzen.

Der KIM wiederholt hier eine Reihe von 16 Tönen. Im Speicherbereich 0300 bis 03BF wird in jeweils zwei aufeinanderfolgenden Bytes die vom Tongenerator des Programmes zum Erzeugen eines Tones benötigte Information (von C_1 bis h^5) gespeichert; z. B. stehen bei 0342 und 0343 die beiden Bytes, die den Generator mit dem Kammerton a oszillieren lassen.

Soll nun eine Tonfolge erklingen, so muß das niedere Byte der ersten Adresse des Tones aus der Maschinencode-Auflistung (*Bild 1*) 0300 bis 03BF in den Speicherbereich von 00D0 bis 00DF geladen werden; jeder Speicherplatz entspricht einem Ton. Wäre der erste Ton der Sequenz ein a, dann belegt man dementsprechend 00D0 mit 42.

Nach Eingabe der 16 Töne fehlen nur noch ein Verstärker oder eine Stereoanlage, um die erzeugte Melodie hörbar zu machen. Dazu werden die Phonoeingänge mit dem Anschluß PA 0 (A-14) des KIM verbunden. Ein Spannungsteiler 100:1 zwischen Computer und Verstärker vermeidet eine Übersteuerung der nachfolgenden Stufe. Wird dann noch eine Verbindung des Timer-Interrupt-Ausganges PB7 (A-15) des 6530 mit dem NMI-Eingang des Prozessors (E-6) erstellt, so kann ein Start des Programmes bei 0000 erfolgen und es erklingt die in 00D0 bis 00DF gespeicherte Tonfolge.

Durch einen Tastendruck auf „+" verkürzt sich die Länge einer Sequenz mit jedem Ton. Bei Betätigung der Taste „AD" vergrößert sich die Dauer der Töne. Taste „0" transponiert die Reihe der 16 Töne mit der nächsten Wiederholung um zwei Halbtöne (große Sekunde) nach unten. Mit „3" kann dieser Vorgang rückgängig gemacht werden. Bei den Tasten „4" und „7" beträgt die Verschiebung fünf Halbtöne (Quarte), bei „8" und „B" sieben Halbtöne (Quinte) nach unten oder oben.

Mit dem Start werden Ein-, Ausgänge und der NMI-Vektor gesetzt. Danach ermittelt der Prozessor durch das Monitor-Unterprogramm GETKEY einen Tastendruck und verändert eventuell die Tonlänge, welche von der Zählzeit des Timers abhängt, indem das in den Timer geladene Byte inkrementiert oder dekrementiert wird.

Vor dem Anfangston der Sequenz prüft das Programm auch noch, ob die Melodie transponiert werden soll. Zeigt z. B. ein Tastendruck auf die „3" eine Erhöhung der Sequenz um eine große Sekunde an, so addiert ein Unterprogramm (0240) die Zahl 4 zu den Tonnummern in 00D0 bis 00DF. Dadurch erklingen dann die um zwei Halbtöne höheren Frequenzen.

Als nächster Schritt im Programm erfolgt eine Decodierung der relevanten Toninformation. Die 16 Bits aus dem Bereich 0300 bis 03EF bestimmen die Länge einer Halbwelle des abgegebenen Rechtecksignals. Die letzten sechs Bit geben an, aus wieviel Zeiteinheiten von je 1024 µs die Halbwelle besteht; die nächsten acht Bits die Anzahl der 4-µs-Einheiten. Das zweite Bit verlängert, falls gesetzt, die Halbwelle um 2 µs, das erste Bit um 1 µs. Außerdem muß zu der aus der Toninformation errechneten Zeitdauer noch der feste Wert von 36 µs hinzugezählt werden. Der Ausgang PA 0 wird nach der Dauer einer Halbwelle invertiert, so daß an ihm eine Rechteckspannung mit der gewünschten Tonfrequenz abgegriffen werden kann.

Der Ton erklingt so lange, bis der Timer auf Null heruntergezählt hat und einen Interrupt auslöst. Daraufhin springt der Prozessor an die Programmstelle, die das Unterprogramm GETKEY aufruft. Er verändert gegebenenfalls Tonhöhe oder Länge, startet den Timer von neuem und erzeugt den nächsten Ton.

Natürlich läßt sich das Programm auf eine andere Sequenzlänge als 16 „umbauen". Dazu gibt man lediglich die um eins verminderte Tonanzahl in hexadezimaler Form in die Speicherzellen 0032 und 0241 ein; allerdings darf die Zahl 1F nicht überschritten werden, da sonst eine Überschneidung mit dem vom Monitorprogramm belegten Bereich stattfände.

```
KIM-SEQUENZER
0000 D8 A9 01 8D 01 17 A9 7F 8D 03 17 A9 FF 85 E2 A9
0010 1B 8D FA 17 A9 00 8D FB 17 85 E5 20 6A 1F C9 12
0020 D0 06 CE 01 02 4C 2F 00 C9 10 D0 03 EE 01 02 A4
0030 E2 C0 0F D0 5B A0 FF 84 E2 C9 03 D0 0B A9 04 8D
0040 46 02 20 40 02 4C 90 00 C9 00 D0 0B A9 FC 8D 46
0050 02 20 40 02 4C 90 00 C9 07 D0 0B A9 0A 8D 46 02
0060 20 40 02 4C 90 00 C9 04 D0 0B A9 F6 8D 46 02 20
0070 40 02 4C 90 00 C9 0B D0 0B A9 0E 8D 46 02 20 40
0080 02 4C 90 00 C9 08 D0 08 A9 F2 8D 46 02 20 40 02
0090 E6 E2 A4 E2 B6 D0 BD 00 03 85 E0 E8 BD 00 03 85
00A0 E1 A9 00 85 E3 85 E4 46 E1 66 E0 90 02 E6 E4 46
00B0 E1 66 E0 90 02 E6 E3 A9 FF 8D 0F 17 4C 00 02 20
00C0 28 24 2C 26 2C 28 28 FF 28 28 28 3C 28 38 28 28
00D0 42 72 68 80 64 76 6E 7C 38 72 68 80 3E 6E 46 76

0200 A9 9E 8D 0F 17 A9 00 C5 E4 D0 04 EA 4C 14 02 A6
0210 00 4C 14 02 C5 E3 F0 01 EA A6 E0 F0 03 CA D0 FD
0220 A4 E1 F0 09 A2 FE CA D0 FD EA 88 D0 F7 EE 00 17
0230 4C 05 02 88 D0 F7 EE 00 17 4C 05 02 3C 3C 3C 1C
0240 A2 0F 18 B5 D0 69 0E 95 D0 CA 10 F6 00

0300 95 3B 3B 38 11 35 14 32 43 2F 99 2C 17 2A B8 27
0310 7B 25 5F 23 60 21 7F 1F B8 1D 0B 1C 76 1A F8 18
0320 8F 17 3B 16 F9 14 CA 13 AB 12 9D 11 9E 10 AD 0F
0330 CA 0E F4 0D 29 0D 6A 0C B6 0B 0B 0B 6B 0A D3 09
0340 44 09 BC 08 3D 08 C5 07 53 07 E8 06 82 06 23 06
0350 C9 05 74 05 23 05 D7 04 90 04 4C 04 0C 04 D0 03
0360 97 03 62 03 2F 03 FF 02 D2 02 A8 02 7F 02 5A 02
0370 36 02 14 02 F4 01 D6 01 BA 01 9F 01 85 01 6E 01
0380 57 01 42 01 2E 01 1B 01 09 01 F8 00 E8 00 D9 00
0390 CB 00 BD 00 B1 00 A5 00 99 00 8F 00 85 00 7B 00
03A0 72 00 6A 00 62 00 5A 00 53 00 4C 00 46 00 40 00
03B0 3B 00 35 00 30 00 2B 00 27 00 23 00 1F 00 1B 00
```

Hexadezimale Programmauflistung. Ab 0300 sind die einzelnen Tonhöhen für C, Cis, D, Dis, E, F, Fis, G, Gis, A, Ais, H, C' usw. zu finden; der Tonumfang erstreckt sich über acht Oktaven

Stichworte zum Inhalt

KIM-1, 6502, 6530, NMI, Interrupt-Timer, Musik.

Dipl.-Phys. Christian Jäkel

Wenn der Speicher überläuft

TI-59-Sparmaßnahmen

Mit maximal 960 Speicherplätzen ist die Kapazität des TI-59 nicht eben knapp bemessen. Entsprechend groß ist die Versuchung, umfangreiche Programme zu schreiben – und dann stößt man eben doch allzu rasch an die Grenzen. Jetzt beginnt der Kampf um die letzten Speicherplätze.

Glücklicherweise kann der TI-59 aber noch etwas mehr als das, was in der Bedienungsanleitung steht. So bleiben immer noch ein paar Lücken, wenn man den Speicher bereits randvoll wähnt. Dieser Aufsatz soll einige Tricks schildern, wie man diese Lücken nutzen kann.

Die H-Register

Für den internen Gebrauch besitzt der TI-59 acht zusätzliche Register, die durch keinen Tastenbefehl direkt erreicht werden können, an die man aber auf Schleichwegen dennoch herankommt. Wir nennen diese Register H1 bis H8. Im Normalbetrieb des Rechners haben sie folgende Funktion:

H1 ist das last-x-Register; rechnet man beispielsweise 25+ 5 = 30, so ist der Inhalt von H1 gleich 25.

H2 bis H6 sind die Speicher für die Klammerebenen; nachdem man gerechnet hat: $12 \cdot (17 + 4) = 252$, ist H1 = 12 und H2 = 17. Die Registerinhalte bleiben nach Abschluß der Rechnung erhalten, bis sie von der nächsten Rechnung überschrieben werden.

H5 bis H8 sind die Druckerregister, die mit OP 01 bis OP 04 geladen werden. Daran erkennt man, daß ab der vierten Klammerebene die Druckerregister überschrieben werden – eine Eigenheit des TI-59, die auch nicht in der Betriebsanleitung steht. Der Befehl 1122334455 OP 01 lädt beispielsweise 0,001122334455 in Register H5. Dies ist die interne Standarddarstellung des alphanumerischen Code; sie ist 13stellig, d.h. die drei Schutzstellen werden mitbenutzt. OP 01 bis 04 laden also H5 bis H8 in Standarddarstellung.

Direkter Zugriff auf die H-Register

Will man mit den H-Registern direkt arbeiten, so werden sie über die Befehle HIR XX angesprochen. HIR hat den Tastencode 82, jedoch gibt es keine Taste dafür. Um also in einem Programm HIR XX zu schreiben, muß man einen Umweg beschreiten: Man tippt STO 82 STO XX und erzeugt damit die Programmzeilen:

42 STO
82 82
42 STO
XX XX

Nun löscht man die beiden STO-Befehle (Taste DEL), und es verbleiben die gewünschten Befehle:

82 HIR
XX XX

Alle H-Register lassen sich direkt laden, nämlich mit HIR 0X, wobei X für die Nummer des Registers steht. So wird z.B. mit 12,5 HIR 03 die Zahl 12,5 in H3 geladen.

Alle H-Register lassen sich direkt abrufen, nämlich mit HIR 1X. Nach vorigem erscheint also mit HIR 13 die 12,5 wieder im x-Register (Anzeige). Alle H-Register haben volle Speicherarithmetik, wobei man allerdings eine anschließend zu besprechende Ausnahme zu beachten hat.

Addition:
y HIR 3X bewirkt HX := HX + y
Multiplikation:
y HIR 4X bewirkt HX := HX · y
Subtraktion:
y HIR 5X bewirkt HX := HX – y
Division:
y HIR 6X bewirkt HX := HX : y

Bei der Division erfolgt eine Fehlermeldung, falls y = 0. Beispiel: 7 HIR 05 3 HIR 35 2 HIR 65 HIR 15 rechnet (7 + 3) : 2 = 5 im Register H5 und läßt das Ergebnis im x-Register erscheinen.

HIR 7X, HIR 8X, HIR 9X sind funktionsgleich mit HIR 6X. Alle Befehle HIR 2X sowie alle Befehle HIR X0 und HIR X9 sind funktionslos (= NOP).

Es ist nun noch die erwähnte Einschränkung zu besprechen. Die Speicherarithmetik der H-Register arbeitet (außer im EE- oder Eng-Format) in der oben beschriebenen Weise nur, solange $y \geqq 1$. Ist nämlich $y < 1$, und zwar von der Form

$y = a \cdot 10^{-x}$ mit $1 \leqq a < 10$,

so wandelt jeder Befehl der Speicherarithmetik zunächst y um in $y' = a \cdot 10^{x}$, um dann die Arithmetik mit y' statt mit y auszuführen. Daher erfolgt auch bei Addition, Multiplikation und Subtraktion eine Fehlermeldung, falls y = 0 ist. Die Befehlsfolge 0,25 HIR 35 ersetzt also zunächst 0,25 durch 25 und addiert 25 zu H5. Auch im x-Register erscheint 25.

Alle H-Register lassen sich nur direkt adressieren; HIR IND XX wird nicht als Befehl angenommen.

Druckertricks

Mittels HIR 05 bis HIR 08 lassen sich die vier Druckerregister in jeder beliebigen nichtstandardisierten Weise laden. Wird danach mittels OP 05 der Ausdruck veranlaßt, so werden die letzten zehn signifikanten Stellen des Registerinhalts einschließlich der drei Schutzstellen als alphanumerischer Code interpretiert, unabhängig von der Position des Dezimalpunktes. Beispielsweise kann man mit 1234567890 + 0.123 = HIR 05 die Zahl

Zehner ↓ / Einer →	0	1	2	3	4	5	6	7	8	9
0		0	1	2	3	4	5	6	7	8
1	7	8	9	A	B	C	D	E	7	8
2	–	F	G	H	I	J	K	L	M	N
3	M	N	O	P	Q	R	S	T	M	N
4	.	U	V	W	X	Y	Z	+	×	★
5	×	★	√	π	e	(	)	,	×	★
6	↑	%	▶	/	=	'	×	x̄	2	?
7	2	?	÷	!	Σ	Δ	Π	Σ	2	?
8		0	1	2	3	4	5	6	7	8
9	7	8	9	A	B	C	D	E	7	8

Bild 1. Symbolcode-Tabelle des TI-59 in erweiterter Form: Auch den mit 8 und 9 gebildeten Codes wird ein Symbol zugeordnet

1234567890.123 in H5 laden. Die letzten zehn signifikanten Stellen sind 4567890123, mit OP 05 wird also Y x̄ 8 0 H gedruckt. Lädt man dagegen das H5-Register mit OP 01, so wird von jeder Zahl nur der Integer-Anteil interpretiert und in Standarddarstellung gespeichert.

Dies eröffnet nun die Möglichkeit, in einer Speicherzelle mehr als 5 alphanumerische Symbole unterzubringen, indem man nämlich den Dezimalpunkt geschickt plaziert und die Schutzstellen mitbelegt. Beispiel: Es stehe in R00 die Zahl 3101.735350135. Dann bewirkt RCL 00 OP 01 OP 05 den Ausdruck von NO (Code 3101), aber mit RCL 00 HIR 05 OP 05 wird ERROR (Code 1735350135) ausgedruckt.

Man beachte, daß die 1 vor dem Dezimalpunkt in diesem Beispiel sowohl als Einerstelle von 01 (Symbol „0") als auch als Zehnerstelle von 17 (Symbol „E") fungiert. Bei dieser Methode passen daher nur solche Symbole aneinander, bei denen die Einerstelle des ersten gleich der Zehnerstelle des zweiten ist. Mit der Symbolcode-Tabelle (*Bild 1*) lassen sich aneinanderpassende Symbole sofort finden: Zu jedem Symbol der Spalte i paßt jedes Symbol der Zeile i; die Symbole in der Hauptdiagonale passen zu sich selbst.

Erweiterung der DSZ-Operation

Der DSZ-Befehl (decrement and skip on zero) wirkt laut Bedienungsanleitung nur auf die Register R00 bis R09. Auch hier gibt es indessen Schleichwege.

Die übliche Form eines DSZ-Befehls ist:

97 DSZ
03 03
11 A

Er dekrementiert den Inhalt von R03 und veranlaßt einen Sprung zur Marke A, solange der Inhalt von R03 ungleich 0 ist. Tippt man nun in einem Programm DSZ STO 23 A, und „deletet" man anschließend den STO-Befehl, so verbleibt:

97 DSZ
23 23
11 A

Der DSZ-Befehl wird dann auf Register R23 angewendet. Auf diese Weise kann man mit DSZ jeden Speicherplatz ansprechen, nicht nur R00 bis R09. Ausgenommen ist R40, denn DSZ 40 wird interpretiert als

97 DSZ
40 IND

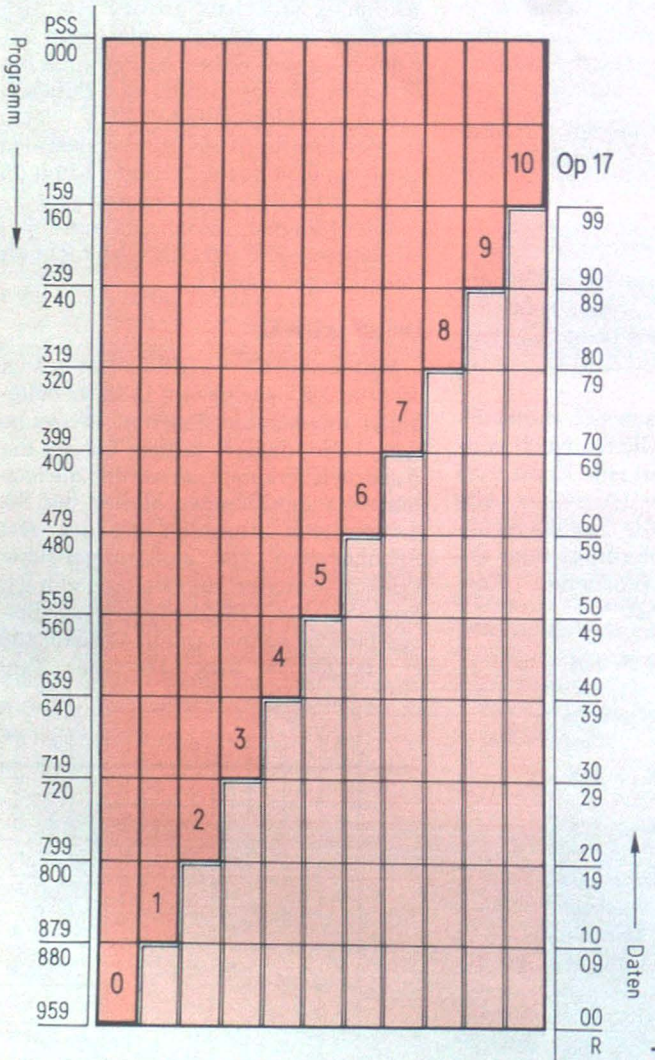


Bild 2. Grob-Splitting des Speicherbereiches mit X OP 17

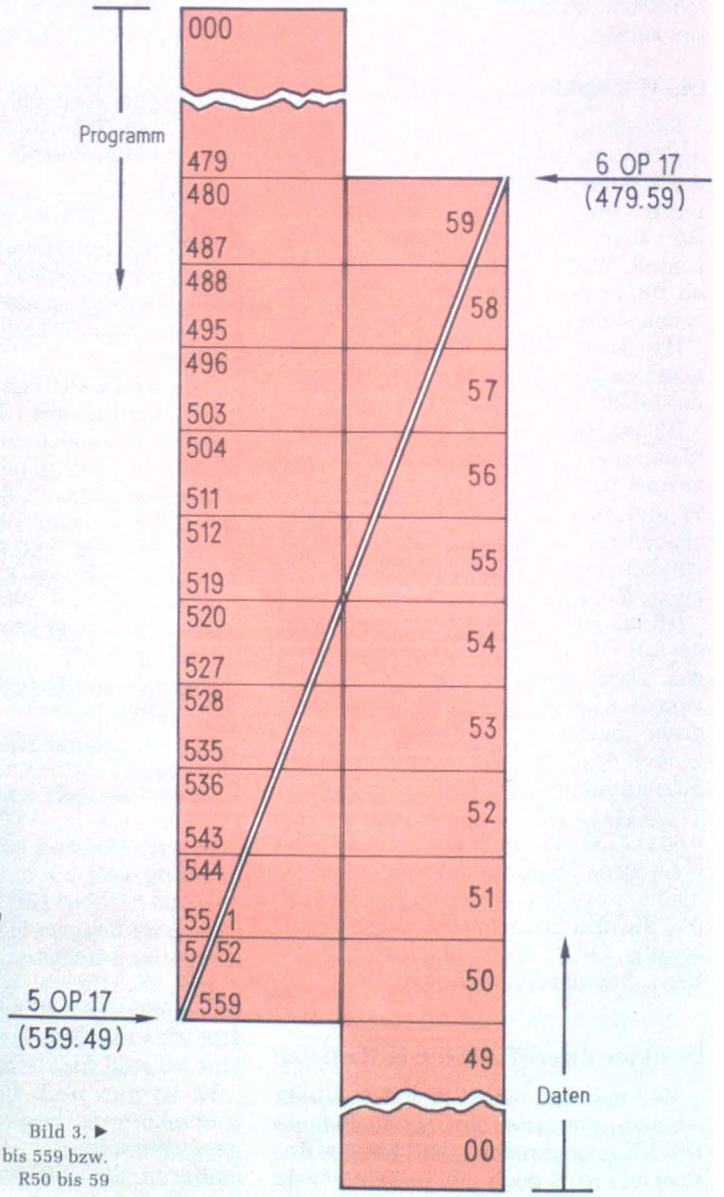


Bild 3. ▶
Fein-Splitting des Speicherbereiches, gezeigt an dem Zehnerblock PSS 480 bis 559 bzw. R50 bis 59

und es muß nun erst die Nummer des Registers kommen, in dem die Nummer des zu bearbeitenden Registers hinterlegt ist, und danach erst die Verzweigungsadresse; andernfalls erfolgt eine Fehlermeldung. Mit

```
97 DSZ
40 IND
15 15
12 B
```

wird also DSZ auf dasjenige Register angewandt, dessen Nummer in R15 steht. Dies darf nun allerdings jeder Speicherplatz sein, auch R40.

Soll als Verzweigungsadresse keine Sprungmarke, sondern eine Programmzeile angegeben werden, so schreibt man die komplette Befehlsfolge zunächst für Register R00:

```
97 DSZ
00 00
04 04 }
73 73 }  Adresse 473
```

Sodann fügt man mittels INS zwischen DSZ und 00 noch einen STO-Befehl ein, dessen Registernummer die 00 überschreibt, und schließlich „deletet" man das STO:

97	DSZ	97	DSZ
42	STO	23	23
23	23	04	04
04	04	73	73
73	73		

Mit diesem Trick kann man also DSZ auf jedes Register anwenden und braucht für solche Zwecke nicht mehr R00 bis R09 freizuhalten.

Speicherbereichs-Splitting

Bekanntlich läßt sich der Speicherbereich des TI-59 mittels X OP 17 auf Programm- und Datenregister aufteilen, und zwar ändern sich die Bereiche in Blöcken von zehn Datenregistern, wobei zehn Datenregister jeweils 80 Programmspeicherplätze ergeben (*Bild 2*).

Bei der Einstellung auf 479.59 hat man 480 Programmspeicherplätze und 60 Datenregister; bei 559.49 sind es 560 Programm- und 50 Datenspeicherplätze. Wie nun, wenn man mehr als 50, aber nicht alle 60 Datenregister braucht, dafür aber mit dem Programm in 480 Schritten nicht auskommt? Hier gibt es die Möglichkeit, den Speicherbereich weiter zu splitten, jeder unbenutzte Datenspeicherplatz liefert acht weitere Programmspeicherplätze. *Bild 3* zeigt dies für das obige Zahlenbeispiel.

Praktisch geht man hierbei z.B. so vor, daß man ab PSS 480 eine Subroutine unterbringt, die die nicht benötigten Datenspeicher belegt. Im Programm ruft man diese Subroutine auf mittels 5 OP 17 SBR 480 6 OP 17. Durch 5 OP 17 werden die Programmspeicherplätze 480 bis 559 lesbar und gleichzeitig die Register R50 bis R59 unzugänglich; mit 6 OP 17 werden die Register R50 bis R59 wieder lesbar und dafür PSS 480 ff unzugänglich.

Man hat lediglich darauf zu achten, daß der komplette Subroutine-Aufruf noch vor PSS 480 steht, daß die Subroutine keine Register ab R50 anspricht, und daß das Hauptprogramm keine Register anspricht, die von der Subroutine belegt sind.

Für andere Speicherbereichsverteilungen kann man analog vorgehen. Mit einer Subroutine geht das Speicher-Splitting besonders einfach, aber bei konsequenter Beachtung der Regeln kann man auch mit dem Hauptprogramm in den gesplitteten Bereich vordringen.

Indem man die Splitting-Regeln bewußt verletzt und mit dem Programm gezielt Speicherzellen anspricht, in denen Programmteile stehen, kann man übrigens das Programm veranlassen, sich selbst zu verändern, Programmteile auszutauschen bzw. zu modifizieren und dergleichen mehr. Dies kann in einigen speziellen Fällen – insbesondere in der Testphase von Programmen – vorteilhaft sein und eröffnet eine Reihe weiterer interessanter Möglichkeiten. Mit Speicherersparnis hat dies indessen nichts mehr zu tun und würde daher den Rahmen dieses Aufsatzes sprengen. Es dürfte aber nach Studium dieses Artikels nicht schwierig sein, selbst entsprechende Experimente anzustellen.

Stichworte zum Inhalt

TI-59, HIR-Befehle, Druckerregister, Klammerregister, DSZ-Operation, Speicherverteilung

PET speichert Maschinenprogramme

Mit einem kleinen Hilfsprogramm ist es möglich, Maschinenprogramme beim PET wie in Basic auf Kassette zu speichern und wieder zu laden. Das *Bild* zeigt das hierfür erforderliche Maschinenprogramm als Data-Zeilen, die mit einer kleinen Basic-Schleife in den Adressenbereich des Kassettenbuffers 2 transferiert werden, so daß man zur Eingabe keinen besonderen 6502-Monitor (z.B. TIM) benötigt.

Der Start des Programmes erfolgt mit SYS(826). Dann kann der Programmname, die Anfangsadresse und die Endadresse – jeweils hexadezimal – eingegeben werden. Hierbei ist zu beachten, daß die Endadresse der letzten zu speichernden Adresse + 1 entspricht.

Zur Funktionsweise ist folgendes zu sagen: Zunächst wird die laufende Primäradresse (in F1) auf 1 gesetzt (für Rekorder 1, entsprechend der 1 in OPEN X,1,Z). Da der Programmname später beginnend mit der Adresse 0030 im BASIC-Input-Buffer abgelegt werden soll, wird die Adresse des laufenden Filenamens (F9, FA) auf 0030 gesetzt.

Mit den Routinen 0346–0350, 0361–036B und 0376–0380 wird der Text „Programmname: „Anfangsadresse (hex):" und „Endadresse (hex):" ausgegeben (JSR FFD2 interpretiert den Akku-Inhalt als ASCII-Zeichen und schreibt dieses auf die momentane CURSOR-Position). Die Routine 0351–0360 holt den Programmnamen vom Schirm und speichert ihn in den Input-Buffer (maximal 17 Zeichen oder bis zum Carriage-Return). In EE wird dann die Länge dieses Filenamens gespeichert. Schließlich werden noch die Anfangsadresse nach F7, F8 und die Endadresse nach E5, E6 gespeichert. Die hierzu verwendete Routine 038F–03A9 holt jeweils zwei ASCII-Zeichen vom Schirm und kombiniert diese zu einem Byte. *Dieter Smode*

```
10 FORK=826TO994:READA:POKEK,A:NEXT
100 DATA169,1,133,241,169,48,133,249,169,0,133,250,162,15,189
110 DATA169,3,32,210,255,202,208,247,32,207,255,201,13,240,7
120 DATA149,48,232,224,17,208,242,134,238,162,23,189,184,3,32
130 DATA210,255,202,208,247,32,143,3,133,248,32,143,3,133,247
140 DATA162,19,189,207,3,32,210,255,202,208,247,32,143,3,133
150 DATA230,32,143,3,133,229,32,177,246,96,32,160,3,10,10
160 DATA10,10,133,88,32,160,3,41,15,5,88,96,32,207,255
170 DATA201,58,144,2,105,8,96,32,58,69,77,65,78,77,77
180 DATA65,82,71,79,82,80,147,32,58,93,88,69,72,91,32
190 DATA69,83,83,69,82,68,65,83,71,78,65,70,78,65,13
200 DATA32,58,93,88,69,72,91,32,69,83,83,69,82,68,65,68,78,69,13
READY.
```

Startet man dieses Basic-Programm, so wird ein Maschinenprogramm ab Adresse 033A (hex) generiert, mit dem man Maschinenprogramme so auf Kassette abspeichern kann, daß sie sich wie Basic-Programme wieder laden lassen

Eindeutig zweideutig

Das Gleichheitszeichen in Basic

Wenn sich Anfänger in die Geheimnisse von Basic einarbeiten, dann erweist sich in aller Regel als eine der Klippen auf dem Weg zum Verständnis dieser Programmiersprache, daß dem Gleichheitszeichen zwei völlig verschiedene Bedeutungen zukommen. Einmal dient dieses Zeichen dazu, einer Variablen einen Wert zuzuweisen – oder anders formuliert: Jene Gruppe von Speicherstellen, für die der Variablenname eine symbolische Bezeichnung ist, mit einem Wert zu laden, der sich rechts vom Gleichheitszeichen befindet – beziehungsweise der sich ergibt, wenn der Rechner die dort befindliche Rechenanweisung ausführt. Beispiele: PI=3.1416 oder aber auch DM=PF/100.

Zum zweiten bereitet das Gleichheitszeichen, im Zusammenwirken mit dem Basic-Wort IF („Wenn..."), einen bedingten Sprung vor. Beispiel:
IF DM=0 THEN PRINT "KEINE MUEDE MARK MEHR IN DER KASSE".
Hier verändert das Gleichheitszeichen den Wert der links von ihm stehenden Variablen in keiner Weise; unseren Kassenbestand DM läßt ein Statement dieser Art durchaus ungeschoren.

Frühe Basicversionen unterscheiden zwischen beiden Fällen säuberlich: Bei Wertzuweisungen hatte dem Gleichheitszeichen das Wort LET voranzugehen: LET A=B, „lasse A den Wert von B annehmen". Eine saubere Trennung zwischen beiden Bedeutungen finden sich auch heute noch in Programmiersprachen wie FOCAL („S A=B", setze A gleich B) und PASCAL („BESTAND:=ALTBESTAND – ENTNAHME", hier macht's der Doppelpunkt hinter dem ersten Variablennamen).

Zurück zu Basic: Auch nach dem Verzicht auf den Zwang, Wertzuweisungen mit LET einzuleiten, blieben die Verhältnisse überschaubar, solange jedem IF („Wenn...") ein THEN („...dann...") zu folgen hatte: Es galt die Regel, Gleichheitszeichen zwischen beiden Schlüsselworten waren als „Gleich" zu lesen, alle anderen als „Sei": A=B also „A sei B", und „IF A=B THEN ..." war zu lesen als „Wenn A gleich B, dann ...". Heute hilft diese einfache Regel nicht mehr weiter. Moderne Basicinterpreter lassen es neuerdings nämlich durchaus zu, in gewissen, häufig vorkommenden Fällen auf das zweite Schlüsselwort THEN zu verzichten. So ist in den meisten Microsoft-Dialekten jüngeren Datums das abgekürzte Statement IF A=B GOSUB 100 als Ersatz der Langform IF A=B THEN GOSUB 100 durchaus erlaubt.

Somit nimmt unsere Regel über die Deutung des Gleichheitszeichens in Basic folgende Gestalt an:
(1) Alle Gleichheitszeichen sind als „Sei" zu lesen.
(2) Ausgenommen hiervon und als „Gleich" zu lesen sind Gleichheitszeichen zwischen den Schlüsselworten IF auf der linken Seite und THEN, GOTO oder GOSUB rechts.
Mit dieser Regel wird man in 999 von 1000 Fällen zurechtkommen. Freilich: Keine Regel ohne Ausnahme. Es gibt einen besonders raffinierten Programmiertrick, in dem die obige Interpretations-Vorschrift versagt – das Beispiel dafür:

```
10 INPUT "ERSTE ZAHL"; A
20 INPUT "ZWEITE ZAHL"; B
30 C = B = A: REM KEIN DRUCKFEHLER!
40 REM HIER DUERFEN BELIEBIGE STATEMENTS FOLGEN, SOFERN DIESE C KEINEN NEUEN WERT ZUWEISEN
100 IF C THEN PRINT "BEIDE ZAHLEN WAREN GLEICH"
200 END
```

Was geschieht in diesem Programm mit der merkwürdigen Zeile 30 und ihrem Doppel-Gleichheitszeichen? Hier müssen wir erst einmal darauf hinweisen, daß moderne Basic-Interpreter außer mit den bekannten Variablenarten wie REAL, INTEGER und STRING intern noch mit einer vierten Variablenart arbeiten, dem Typ „BOOLEAN" nämlich, der – im Unterschied zur Sprache PASCAL zum Beispiel, wo diese Variablenklasse dem Programmierer unmittelbar zur Verfügung steht – der in BASIC nur über Tricks zugänglich ist. BOOLEAN-Variable sind dadurch gekennzeichnet, daß sie nur zwei Werte annehmen können, die man „WAHR" und „UNWAHR" (in Pascal: TRUE und FALSE) nennt.

Um nun „WAHR" und „UNWAHR" abbilden zu können, verschlüsseln moderne Basic-Interpreter beide Begriffe durch Zahlen. Die Rechner ITT 2020 und Maschinen der Apple-Klasse setzen für WAHR die Zahl 1, für UNWAHR die Zahl 0; andere Rechner benutzen negative Zahlen. Im IF-Teil einer Anweisung ermittelt der Rechner nun den Wert WAHR oder UNWAHR für die Bedingung und speichert sie; den THEN-Teil führt er nur aus, wenn er in der internen BOOLEAN-Variable den Schlüssel für „WAHR" findet. Zurück in unser kleines Musterprogramm. In Zeile 30 vergleicht der Rechner die Zahlen A und B, von denen durch das Gleichheitszeichen ja behauptet wird, sie seien identisch. Erkennt der Rechner diese Behauptung als wahr, dann weist er der Variablen C den BOOLEAN-Schlüsselwert für „WAHR" zu, andernfalls den Wert für „UNWAHR". C bleibt nun unverändert und wird erst wieder in Zeile 100 aufgerufen, die der Computer so interpretiert: „Wenn in C der Wert WAHR steht, dann – und nur dann – drucke aus: Beide Zahlen waren gleich."

Nebenbei bemerkt: Wer wissen möchte, wie sein Rechner mit der Wahrheit umgeht, der kann das Musterprogramm noch um eine Zeile erweitern und dann mit verschiedenen Eingabewerten für A und B experimentieren – die Zeile lautet wie folgt:

```
120 PRINT" === SCHLUESSEL FUER C: ";C:GOTO 10
```

Leser, die ein wenig mit diesem Demonstrationsprogramm spielen und dabei das zweite Gleichheitszeichen in Zeile 30 durch Vergleichsoperatoren wie Ungleich, Kleiner, Größer-Gleich ersetzen und den Ausgabetext der Zeile 100 entsprechend ändern, die werden in Kürze eine Art „Sprachgefühl" in Sachen Basic-Gleichheitszeichen entwickeln. Ein letztes Beispiel diene als Anregung für eigene Versuche des Lesers:

```
 10 INPUT "A"; A
 20 INPUT "B"; B
 30 INPUT "C"; C
 40 INPUT "D"; D
 50 P1=A=B
 60 P2=B=C
 70 P3=C=D
 80 P4=D=A
 90 IF (P1 AND P2 AND P3 AND P4) THEN PRINT "ALLE ZAHLEN GLEICH.-"
100 IF NOT (P1 OR P2 OR P3 OR P4) THEN PRINT "KEINE AUFEINANDER FOLGENDEN PAARE.-"
110 GOTO 10
```

Man sieht: Mutiger Einsatz des Gleichheitszeichens und die Benutzung der in BASIC offiziell gar nicht existierenden Variablenklasse BOOLEAN (benannt nach dem Mathematiker George Boole, dem Schöpfer eines Systems zur abstrakten Formulierung logischer Fragestellungen; 1815-1864) gibt das Werkzeug zu eleganten Programmierlösungen an die Hand. *Hans-Georg Joepgen*

Heinz J. Schilling

Datenspeicherung beim KIM-Basic

Ähnlich wie beim AIM-65 tritt auch beim KIM-1-Basic das Problem auf, daß man mit SAVE und LOAD zwar Programme, nicht aber Daten sichern und laden kann. Nachstehend wird eine Basic-Routine beschrieben, die dieses Problem beim KIM-1 löst.

Datentypen und Namen

Das KIM-Basic kennt folgende Datentypen:
- einfache Variable (Gleitkomma-Darstellung)
- Ganzzahl-Variable (Integer)
- numerische Felder (Gleitkomma-Darstellung)
- Integer-Felder
- Stringvariablen

Die Variablennamen können z.B. A, AA, A1 o.ä. sein. Durch Anhängen von % wird ein Integer-Name, durch Anhängen von $ ein String-Name gekennzeichnet. Intern besteht jeder Name aus zwei Bytes, wobei das zweite Null ist, wenn der externe Name nur ein Zeichen lang ist. Bei String-Namen wird das höchstwertige Bit des zweiten Byte auf 1 gesetzt, bei Integer-Namen geschieht das bei beiden Bytes. So wird etwa der Name A als 4100, A1 als 4131, I% als C980 und S$ als 5380 abgespeichert.

Symbol-Listen

Es gibt eine Symbolliste für Variable und eine für Felder (Matrizen). Sie werden hinter dem eigentlichen Basic-Programm abgespeichert. Am Ende des verfügbaren bzw. definierten Speicherbereichs liegen die Strings. Das Microsoft-Basic hat in der Zero-Page eine Reihe von Zeigern auf diese Listenbereiche:

78, 79 Programm-Anfangsadresse
7A, 7B Variablen-Symbolliste, Anfangsadresse
7C, 7D Matrizen-Symbolliste, Anfangsadresse
7E, 7F Endadresse der Matrizen-Symbolliste
80, 81 Anfangsadresse des Stringbereichs
84, 85 Endadresse des Stringbereichs

Interndarstellung von Variablen

Variable belegen sieben Bytes in der Variablen-Symbolliste, und zwar zwei Bytes für den Namen und fünf Bytes für den Wert. Bei Integer-Variablen sind die letzten drei Bytes nicht belegt, da ihr Wert nur 16 bit lang ist.

Felder belegen Platz in der Matrizen-Symbolliste, der sich wie folgt ergibt: Zwei Bytes enthalten den Namen, zwei Bytes geben die Länge des Eintrags an (L, H), ein Byte gibt die Anzahl der Dimensionen der Felder an. Pro Dimension werden zwei Bytes mit der Zahl der Elemente in dieser Dimension benötigt (H, L). Außerdem muß natürlich noch Platz für die Daten selbst vorhanden sein. Strings, die mittels DIM definiert sind, haben zwar in der Feldliste einen Eintrag, er wird aber nur zur Überprüfung der Dimension verwendet. Der eigentliche Eintrag erfolgt in der Variablen-Symbolliste mit einem Element folgenden Inhalts: Zwei Bytes für den Namen, ein Byte String-Länge (max. 255), zwei Bytes mit der Anfangsadresse (L, H) des Strings im String-Bereich. Die restlichen zwei Bytes werden hier nicht benutzt. Der Verweis ändert sich bei jeder Operation, durch die der String beeinflußt wurde, da der jeweils neue String an einer anderen Stelle liegt. Aus *Bild 1* gehen die Variablendefinitionen für unsere Save- und Load-Routinen hervor.

Die Save- und Load-Routinen

Die in *Bild 2* wiedergegebenen Programmteile basieren auf den oben angegebenen Variablendarstellungen und Adressen. Der Dateiname wird entsprechend Bild 1 in die Variable TN eingetragen.

Es wird hier vorausgesetzt, daß die sowieso für das Laden und Sichern des Basic-Programms erforderlichen Tape-Routinen als Unterprogramme vorhanden sind und nicht etwa, wie beim KIM-ROM-Monitor, keinen RTS-Rücksprungbefehl enthalten. Die im Beispiel verwendeten Routinen sind Hypertape (Start 0503) und Hyperload (Start 0506). Wegen der relativ geringen Geschwindigkeit selbst bei Hypertape verbietet es sich, von wenigen Aus-

```
100 REM VARIABLEN FUER TAPE-ROUTINEN
110 TS=0:TE=0: REM FUER ABSUCHEN DER SYMBOLLISTE
120 TL=0:TH=0: REM FUER ADRESSEN DES TAPE-BETRIEBS
130 TN=0:TV=0: REM FUER SYMBOL UND NAMENSVERGLEICH
140 TT=0:TR=0: REM FUER DATENTYP UND TRANSPORTRICHTUNG
150 :
160 :
170 REM BEISPIEL FUER EINTRAG VON SYMBOLNAMEN IN TN
180 REM ============================================
190 TN=ASC("F")*256:              REM NAMEN DES FELDES F(20)
200 TN=ASC("I")*256+ASC("2"):     REM NAMEN DER VARIABLEN I2
210 TN=(ASC("I")+128)*256+ASC("0")+128: REM NAMEN DER VARIABLEN I0%
220 TN=ASC("S")*256+ASC("1")+128):REM NAMEN DES STRINGS S1$
230 :
240 :
500 DIM N$(50),S$(50),F(20): I2=0: I0%=0
510 N$=" ": FOR I=2 TO 50: N$=LEFT$(N$,I-1)+" ": NEXT I: REM NULLSTRING
520 :
540 REM BEISPIEL FUER EINLESEN EINES STRINGS
550 REM =====================================
560 S$=N$: REM STRING AUF VOLLE LAENGE BRINGEN DURCH NULLSTRING
570 TN=ASC("S")*256+128: REM STRINGNAMEN EINTRAGEN
580 TT=0: TR=1: REM "STRING" UND "LOAD" EINTRAGEN
590 GOSUB 30000: REM AUFRUF DER I/O-ROUTINE
600 :
610 :
OK
```

Bild 1. Verarbeitung der Variablen, die bei den Kassettenroutinen verwendet werden

nahmefällen abgesehen, einzelne Variablen zu transportieren; besser ist es, Variable in ein numerisches Feld zu übertragen und dieses dann zu transportieren.

Die Basic-Routine von Bild 2 kann auf Band abgespeichert und an ein vorhandenes Programm angefügt werden, wenn ein weiterer Basic-Kniff benutzt wird.

Das Laden unterteilter Basic-Programme

Der Basic-LOAD-Befehl orientiert sich beim Laden an der Anfangsadresse in hex 78/79. Man kann nun an einen schon geladenen Programmteil einen weiteren hinzufügen, indem man die Anfangsadresse mit POKE-Befehlen verändert. Dazu liest man die Endadresse des Programms aus hex 7A/7B aus, vermindert sie um 3 (so lange ist der END-Befehl, auch wenn er nicht extra eingetragen wurde) und trägt sie in die Adressen hex 78/79 ein. Nach dem Laden des zusätzlichen Programmteils muß man allerdings die ursprüngliche Anfangsadresse in 78/79 eintragen, um den richtigen Start des Programms zu ermöglichen:

```
LOAD
OK
PRINT PEEK(120);PEEK(121);PEEK
(122);PEEK(123)
66 64 01 67
POKE 120,254:POKE 121,66
LOAD
OK
POKE 120,66:POKE121,64
```

Man beachte, daß beim Subtrahieren von 3 der Übertrag richtig beachtet wird. Die Adressen liegen in der Reihenfolge L/H vor. Aus der Adresse 67/01 wurde durch Subtraktion von 3 die neue Anfangsadresse 66/254, da das L-Byte nur bis 255 zählt!

Selbstverständlich ist die beschriebene Methode nur anwendbar, wenn die aneinandergehängten Programmteile mit aufsteigenden Zeilennummern vorliegen. Deshalb wurden auch für die Save- und Load-Routinen in Bild 1 Zeilennummern ab 30 000 belegt.

```
30000 REM SUBROUTINE FUER DATEN-SAVE/LOAD
30010 REM ================================
30020 REM    TT = 0    STRING
30030 REM       = 1    VARIABLE
30040 REM       = 2    FELD
30050 REM    TR = 0    SAVE
30060 REM       = 1    LOAD
30070 REM    TN = NAMEN DER VARIABLEN/DES FELDES/ DES STRINGS
30080 IF TT>1 THEN 30340
30090 TS=PEEK(123)*256+PEEK(122): REM AA SYMBOLLISTE VAR/STRING
30100 TE=PEEK(125)*256+PEEK(124): REM EA SYMBOLLISTE
30110 TV=PEEK(TS)*256+PEEK(TS+1): REM NAMEN IN SYMBOLLISTE
30120 IF TV=TN THEN 30150: REM EINTRAG GEFUNDEN
30130 TS=TS+7: IF TS<TE THEN 30110
30140 PRINT"EINTRAG NICHT GEFUNDEN": STOP
30150 IF TT=1 THEN 30290
30160 TL=PEEK(TS+3): TH=PEEK(TS+4): REM AA STRING
30170 POKE 6133,TL: POKE 6134,TH: REM SAL,SAH
30180 TL=TL+PEEK(TS+2): REM EA = AA + LNG
30190 IF TL>255 THEN TL=TL-256: TH=TH+1: REM CARRY BEACHTEN
30200 POKE 6135,TL: POKE 6136,TH: REM EAL,EAH
30210 IF TR=1 THEN 30250
30220 POKE 6137,65: REM ID=$41 FUER SAVE
30230 POKE 8256,6: POKE 8257,5: REM USRLOC HYPERTAPE = $0506
30240 GOTO 30270
30250 POKE 6137,255: REM ID=$FF FUER LOAD
30260 POKE 8256,3: POKE 8257,5: REM USRLOC HYPERLOAD = $0503
30270 TE=USR(0): REM AUFRUF HYPERTAPE/HYPERLOAD
30280 RETURN
30290 TL=TS-INT(TS/256)*256: REM AA VARIABLE
30300 TH=INT(TS/256)
30310 POKE 6133,TL: POKE 6134,TH: REM SAL,SAH
30320 TL=TL+7: REM EA = AA + LNG
30330 GOTO 30190
30340 TS=PEEK(125)*256+PEEK(124): REM AA SYMBOLLISTE FELD
30350 TE=PEEK(127)*256+PEEK(126): REM EA SYMBOLLISTE
30360 TV=PEEK(TS)*256+PEEK(TS+1): REM NAMEN IN SYMBOLLISTE
30370 IF TV=TN THEN 30400: REM EINTRAG GEFUNDEN
30380 TS=TS+PEEK(TS+3)*256+PEEK(TS+2): IF TS<TE THEN 30360
30390 GOTO 30140
30400 TL=TS-INT(TS/256)*256: REM AA FELD
30410 TH=INT(TS/256)
30420 POKE 6133,TL: POKE 6134,TH: REM SAL,SAH
30430 TL=TL+PEEK(TS+2): TH=TH+PEEK(TS+3): REM EA = AA + LNG
30440 GOTO 30190
OK
```

Bild 2. Was dieses Programm tut, läßt sich durch vorheriges Setzen bestimmter „Steuervariablen" bestimmen

µC-Nachrichten

Für den TRS-80 ist nun Software auch auf Disketten erhältlich – eine Ergänzung der Kassetten-Software von Instant Software Inc. (ISI). Den Deutschland-Vertrieb hat der Micro-Shop Bodensee in Markdorf. Derzeit steht ein Finanz-Paket zur Verfügung, es sollen Programme zur Energieberechnung, für den Geografie-Selbstunterricht und für statistische Zwecke folgen.

□

Das nach eigenen Angaben größte Vertriebsnetz für Personal-Computer-Software besitzt das von Peter Jennings (Microchess!) und Daniel Fylstra 1977 gegründete Unternehmen „Personal Software Inc.". Für 1980 erwarten die elf Angestellten einen Verkaufswert von 3 Mio. Dollar. Die Schwerpunkte liegen auf Programmen für die Computer Apple-II, PET-2001, CBM und TRS-80.

□

Die Münchener Heninger Micro-Computer GmbH ist in etwas größere Räume in der Landwehrstraße 39 umgezogen und vertreibt nun auch den „Superbrain"-Computer, der etwa dem WH-89 von Heathkit entspricht und mit zwei Z-80-Prozessoren arbeitet.

□

Die Stiftung Warentest (Postfach 4141, 1 Berlin 30) prüfte vor kurzem einfache Taschenrechner für den Schulgebrauch. Von den 21 Rechnern mußten immerhin fünf mit mangelhaft zensiert werden – oft eine Folge zu ungenauen Rechnens. Kein einziger erreichte das Qualitätsurteil „sehr gut".

□

Einen „verdeutschten PET" wünscht sich so mancher, der des Englischen nicht ganz mächtig ist. Martin Görlitz (Postfach 852, 5400 Koblenz) hat ein neues Betriebssystem-ROM entwickelt, das den Dialog in deutscher Sprache ermöglicht.

Dipl.-Ing. Ekkehard Flögel

Ein Drucker an zwei Computern

Während der Preis für Mikroprozessor-Systeme relativ niedrig ist, sind die Kosten für Peripherie-Geräte wie Drucker, Stanzer und Floppy-Disk-Laufwerke doch noch sehr hoch. Beim Betrieb von mehreren Prozessoren oder Minicomputern ist es deshalb wünschenswert, diese wahlweise auf verschiedene Ein- und Ausgabegeräte schalten zu können.

Dabei entsteht allerdings ein erheblicher Schaltungsaufwand, denn es müssen neben den acht meist bidirektionalen Datenleitungen noch einige Steuerleitungen umgeschaltet werden. Durch Verwendung des CMOS-Bausteins CD 4529 (MC 14529) läßt sich der Schaltungsaufwand in Grenzen halten. Dieser integrierte Schaltkreis ist ein 2 x 4- oder 8 x 1-auf-1-Multiplexer/Demultiplexer (*Bild 1*). Das Besondere an diesem Baustein ist, daß Ein- und Ausgänge vertauscht werden können. Man kann die Eingangssignale X0...3 auf den Ausgang X durchschalten oder ein Eingangssignal von X an die 4 Ausgänge X0...3. Die Anwahl der Ein- bzw. Ausgänge erfolgt über die Adressenleitungen A und B und über die Steuerleitungen STX bzw. STY. Durch das bidirektionale Verhalten des Bausteins ergeben sich nun interessante Verwendungsmöglichkeiten. Dies soll an zwei Beispielen gezeigt werden.

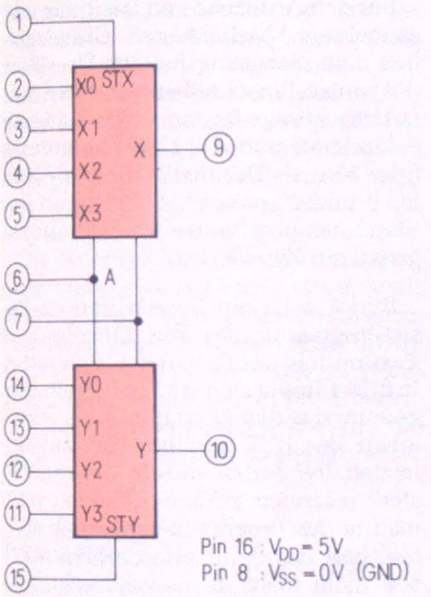


Bild 1. Blockschaltbild des CMOS-Bausteins 4529

Beispiel 1:

Ein Prozessor P soll über sein Tor mit 4 Peripherie-Geräten A, B, C und D verbunden werden. Die gleiche Schaltung gilt auch für 4 Prozessoren, die auf ein Peripherie-Gerät arbeiten. Das Blockschaltbild zeigt *Bild 2*. Mit den beiden Schaltern S1, S2 werden die Eingänge angewählt, die beiden Steuerleitungen STX und STY liegen an +Vdd. Mit acht Bausteinen 4529 erhält man so einen Kreuzschienenverteiler mit 4x16 auf 1x16 Leitungen.

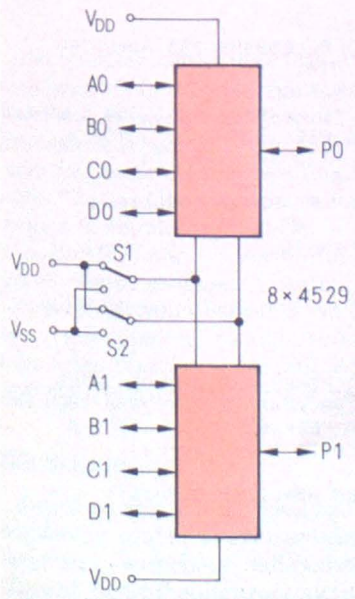


Bild 2. Blockschaltbild zu Beispiel 1

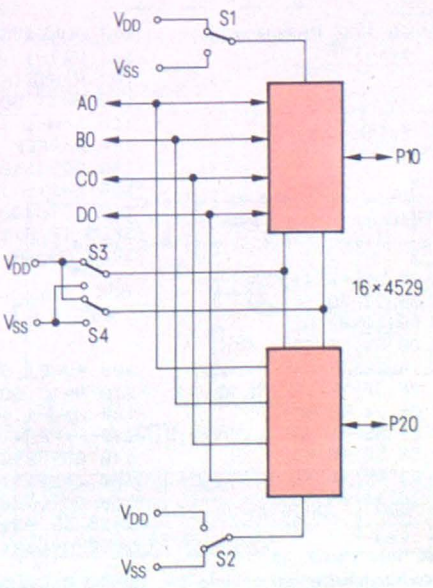


Bild 3. Blockschaltbild zu Beispiel 2

Beispiel 2

Zwei Prozessoren P1 und P2 sollen mit 3 Peripherie-Geräten und untereinander Daten austauschen können. Diese Schaltung erfordert 16 Bausteine 4529 und ist in *Bild 3* dargestellt. Über die Schalter S1 und S2 und die Steuerleitungen STX und STY werden die Prozessoren, über die Schalter S3, S4 und die Adreßleitungen A und B die Peripherie-Geräte angewählt. Beide Prozessoren können über die Leitung D untereinander Daten austauschen.

Aufbau der Schaltung

Zum Aufbau der Schaltung wurde zuerst mit Rechnerunterstützung eine zweiseitige gedruckte Schaltung entworfen. Diese enthielt aber zu viele Durchkontaktierungspunkte und war deshalb für den Laboraufbau nur schwer verwendbar. Es wurde dann eine Grundplatine entworfen, auf der die Verdrahtung mittels Wire-wrapping ausgeführt wurde. Es hat sich gezeigt, daß diese Art von Verdrahtung besonders sicher und irrtumsfrei ist.

Auflösung der Quizfrage aus Heft 9

Welche zwei Bytes durfte man im Beitrag „AIM-65 empfängt Morsezeichen" durch hex EA (NOP-Befehle) ersetzen? Gemeint war der Befehl BMI CW an der Adresse 02A7. Da der Akku bei der Rückkehr aus dem Unterprogramm SHIFT nie gleich hex 15 ist, wird ersatzweise der nachfolgende Sprungbefehl BNE CW ausgeführt, was zum gleichen Sprungziel führt. Haben Sie's gewußt? Wenn nein: Die Frage war ja auch ausgesprochen hinterlistig!

PET-2001 steuert Schreibmaschine

Nachfolgend wird für das in Heft 4/1980 veröffentlichte Schreibmaschinen-Interface die für den Mikrocomputer PET-2001 notwendige Steuersoftware vorgestellt. Sie besteht aus einem Maschinenprogramm- und einem Basic-Teil.

Das Maschinenprogramm *(Bild 1)* ähnelt dem bereits in Heft 4 abgedruckten 6502-Programm, die Änderungen betreffen lediglich die Adressen des I/O-Ports und die Parameter-Übergabe von Basic her. Es findet im Kassettenbuffer 2 ab der Adresse 033A Platz und benutzt eine Umwandlungstabelle, die bei 037D beginnt. Das elektrische Schreibmaschinen-Interface ist an den User-Port anzuschließen; die Datenleitungen D0...D5 an PA 0...5, und die D-Eingänge der Decoder-ICs an PA 7. Die Monoflops werden nicht benützt, und der Ausgang des nun freien NAND-Gatters vor den D-Eingängen ist abzutrennen, wie das schon bei der 6502-Routine in Heft 4 beschrieben wurde. *Bild 2* gibt die Routine nebst Codetabelle als Hex-Dump (dez. Adr.) wieder.

```
037D          PASS 1
037D          PASS 2
0000
0000          ;TYPEWRITER ROUTINE,PA0-5=KEY MATRIX,PA7=NEG STROBE
0000          ;COPYRIGHT H.FEICHTINGER
0000          ;DEVELOPPED WITH AN AIM-65
0000          ;PET-2001 EQUATIONS
0000 CONV             =$D0A7
0000 LSB              =$B4          ;INTEGER ACCU
0000 PA               =$E84F        ;PAD MUST BE SET BY BASIC
0000 FLG              =$3FF
0000                  *=$33A        ;TAPE BUF2
033A
033A          ;PRINT ASCII CHARACTER
033A TYPE  20A7D0 JSR CONV      ;FLOATING PNT TO INTEGER
033D       A4B4   LDY LSB
033F       B97D03 LDA TAB,Y     ;GET KEY CODE
0342       48     PHA
0343       2940   AND #$40      ;ISOLATE SHIFT
0345       CDFF03 CMP FLG       ;SAME SHIFT?
0348       F011   BEQ SAME
034A       AEFF03 LDX FLG       ;NO,TEST FLG
034D       8DFF03 STA FLG       ;STORE NEW
0350       F004   BEQ LOCK
0352       A936   LDA #$36      ;SHIFT CODE
0354       D002   BNE SHIFT
0356 LOCK  A933   LDA #$33      ;SHIFT LOCK CODE
0358 SHIFT 205C03 JSR KEY
035B SAME  68     PLA
035C       ;SUB TO PRESS 1 KEY
035C KEY   A208   LDX #8        ;DURATION
035E       48     PHA
035F       293F   AND #$3F
0361       206D03 JSR KEY1
0364       A214   LDX #$14      ;SHORT DELAY
0366       68     PLA
0367       1002   BPL *+4
0369       A235   LDX #$35      ;LONG DELAY
036B       A9FF   LDA #$FF
036D KEY1  A000   LDY #0
036F KEY2  8D4FE8 STA PA        ;CODE OUT
0372       207B03 JSR RTN       ;WASTE 12 CYCLES
0375       88     DEY
0376       D0F7   BNE KEY2
0378       CA     DEX
0379       D0F4   BNE KEY2      ;DELAY LOOP
037B RTN   60     RTS
037C       00     BRK           ;RESERVED FOR PATCH
037D TAB
037D          ;CODE TABLE
037D          ;TAB-MSB=1: LONG DEL
037D          ;D6=1: SHIFT CHAR
037D                .END
037D           ERRORS= 0000
```

Bild 1. Assembler-Listing der Maschinencode-Routine für die Ausgabe eines Zeichens an das Schreibmaschinen-Interface. Es entstand mit einem AIM-65 nebst Silent-700-Terminal

```
826 20 A7 D0 A4 B4 B9 7D 03 48 29 40 CD FF 03 F0 11
842 AE FF 03 8D FF 03 F0 04 A9 36 D0 02 A9 33 20 5C
858 03 68 A2 08 48 29 3F 20 6D 03 A2 14 68 10 02 A2
874 35 A9 FF A0 00 8D 4F E8 20 7B 03 88 D0 F7 CA D0
890 F4 60 00 3F 3F 3F 3F 3F 3F 3F C6 34 3F 3F 3F
906 B5 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F
922 3F 3F 3F 34 43 70 6C 69 60 6B 44 58 6A 24 08 0C
938 04 03 48 09 31 30 2C 20 2B 18 2A 10 29 68 71 0A
954 6C 1D 4C 2E 6E 53 5B 65 61 5A 5D 52 56 55 4A 4D
970 4B 54 49 4E 72 66 62 59 51 5C 6D 64 63 5E 45 42
986 34 40 28 34 2E 13 1B 25 21 1A 1D 12 16 15 0A 0D
1002 0B 14 09 0E 32 26 22 19 11 1C 2D 24 27 1E 05 02
```

Bild 2. Hex-Dump von Routine und Codetabelle. Die links stehenden Adressen sind dezimal ausgedruckt. Zur Eingabe empfiehlt sich die Verwendung des für den PET-2001 lieferbaren TIM-Monitorprogramms

Für den Ausdruck von Basic aus gibt es mehrere Möglichkeiten. Grundsätzlich muß man zunächst den User-Port (PA) durch Einschreiben von 255 in das Datenrichtungs-Register als Ausgang deklarieren und die USR-Startadresse (hier 33a) als Dezimalwerte in die Zellen 1 und 2 „poken". Jeder Aufruf des Maschinenprogramms mit USR druckt genau ein Zeichen aus.

Bild 3 zeigt ein Demonstrations-Basic-Programm, das die Eingabe von Text mittels des GET-Befehls gestattet. In *Bild 4* findet sich schließlich ein Programm, das den gesamten Bildschirminhalt des PET ausgibt – bis auf die letzten drei Zeilen, die als „Befehlsfenster" reserviert sind. Schließlich muß man ja das Programm irgendwie starten, und das dafür erforderliche RUN soll nicht mehr mit ausgedruckt werden.

Fe.

```
100 POKE1,58:POKE2,3:POKE59459,255:AD=32768
101 GETA$:IFA$=""GOTO101
102 A=ASC(A$):GOSUB1025:GOTO101
110 A=13:GOSUB1025:Z=0:W=40:IF AD>33650 THEN END
120 W=W-1:IF W>0 AND PEEK(AD+W)=32 GOTO120
130 A=PEEK(AD+Z):IF A<32 THEN A=A+64
140 GOSUB1025
150 Z=Z+1:IF Z>W THEN AD=AD+40:GOTO110
160 GOTO130
1025 IF A>128 THEN A=A-96
1030 A=USR(A):RETURN
```

Bild 3. Einfaches Programm zum Schreiben mit der PET-Tastatur

```
100 POKE1,58:POKE2,3:POKE59459,255:AD=32768
110 A=13:GOSUB1025:Z=0:W=40:IF AD>33650 THEN END
120 W=W-1:IF W>0 AND PEEK(AD+W)=32 GOTO120
130 A=PEEK(AD+Z):IF A<32 THEN A=A+64
140 GOSUB1025
150 Z=Z+1:IF Z>W THEN AD=AD+40:GOTO110
160 GOTO130
1025 IF A>128 THEN A=A-96
1030 A=USR(A):RETURN
```

Bild 4. Dieses Basic-Programm druckt zusammen mit dem Maschinenprogramm in Bild 2 den PET-Bildschirminhalt aus

Harald Krake

ZETBUG – ein komfortabler Z-80-Monitor

ZETBUG ist ein leistungsfähiger System-Monitor zum Erstellen und Austesten von Z-80-Maschinenprogrammen. Die hier beschriebene Version läuft direkt auf dem derzeit wohl bekanntesten Z-80-Personal-Computer, dem TRS-80, ist aber grundsätzlich auch auf jedes andere Z-80-System übertragbar.

Geladen wird ZETBUG normalerweise von der Kassette, und zwar mit dem Level-II-System-Kommando:

```
>SYSTEM
*? ZETBUG
*? /
```

Anschließend meldet sich ZETBUG und druckt sein Promptsymbol #, das Zeichen, daß eine Befehlszeile eingegeben werden kann.

```
ZETBUG Z80-MONITOR V1.0
# –
```

Jede Befehlszeile hat die Form:

x aaaa bbbb cccc Enter

wobei x das aus einem ASCII-Zeichen bestehende Kommando, aaaa das erste maximal vierstellige Argument (hexadezimal), bbbb das zweite und cccc das dritte Argument bedeutet. Hinter jedem Argument muß mindestens ein Blank (Leerzeichen) stehen. Zwischen x und aaaa kann, aber muß keines stehen. Nach Drücken der Enter-Taste analysiert ZETBUG die eingegebene Zeile und führt das entsprechende Kommando aus. Wird die Eingabe nicht verstanden, so quittiert ZETBUG dies mit einem erstaunten WHAT? # –.

Im Gegensatz zu manchen anderen Monitorsystemen erfolgt der Dialog mit dem Benutzer daher nicht über ein „live keyboard", d. h. es wird nicht jede gedrückte Taste einzeln interpretiert, sondern im sogenannten „terminal mode", indem die eingegebene Zeile als Ganzes betrachtet wird. Das hat den Vorteil, daß jederzeit noch Korrekturen sowie Cursorpositionierungen möglich sind, bevor man den Input mit Enter abschließt. *Bild 1* zeigt das komplette Maschinencode-Listing.

ZETBUG-Kommandos

 # I

(Initialize Register Save Area (RSA) and Memory-map)

Die RSA ist ein bestimmter Speicherbereich im RAM, der dazu dient, die Z-80-CPU-Register (SP/PC/IX/IY/HL/DE/BC/AF/HL'/DE'/BC'/AF') zwischenzuspeichern, während ZETBUG aktiv ist, d. h. wenn er gerade einen Befehl abarbeitet oder einfach nur auf eine Eingabe wartet. Jedesmal, wenn ein Z-80-Maschinenprogramm zu ZETBUG zurückkehrt, werden sämtliche oben aufgeführten Register in die RSA „gerettet", damit sie nicht verlorengehen. Das umgekehrte Prinzip gilt für die Programmausführung (siehe E). Das I-Kommando setzt nun alle Register auf 0000 sowie den Stackpointer auf 5000 und initialisiert den ZETBUG-Arbeitsbereich, die Memory-map. Außerdem wird der Breakpoint (siehe B) auf 0000, d. h. in den ROM-Bereich gelegt.

 # T aaaa bbbb

(Tabulate memory on screen)

Dieses Kommando druckt den Speicherbereich ab der Adresse aaaa bis zur Adresse bbbb auf dem Bildschirm aus. Am Anfang jeder Zeile steht die jeweilige Speicheradresse, danach folgen 16 Datenbytes. Auf diese Weise kann man sich schnell einen Überblick verschaffen bzw. einen Hexdump erstellen.

Anmerkung:

Bei jedem der 16 ZETBUG-Kommandos können die Argumente weggelassen werden. An deren Stelle muß dann allerdings ein „:" stehen. Dieser veranlaßt, daß die zuletzt verwendeten Argumente, welche in der Memory-map zwischengespeichert sind, für den jeweiligen Befehl aktualisiert werden, ohne daß diese erneut eingegeben werden müssen. Fehlt der „:", dann wird für alle drei Argumente 0000 angenommen. Beispiel: # T: druckt den Speicherbereich ab der Adresse, die zuletzt als Argument 1 (aaaa) eingegeben wurde, bis zur Adresse bbbb, die zuletzt als Argument 2 eingegeben wurde.

 # M aaaa

(Modify and inspect memory)

Bewirkt Übergang in den Modify-Modus. ZETBUG meldet sich mit der Speicheradresse aaaa und dem dazugehörigen Byte. Im Anschluß an das Promptsymbol kann man dann das neue Datenbyte oder auch mehrere, getrennt durch ein Blank, für die nachfolgenden Speicherzellen eingeben. Nach dem Drücken der Enter-Taste werden die neuen Daten in den Speicher eingeschrieben und geprüft, ob nicht gerade im ROM-Bereich oder eine unbelegte Adresse modifiziert werden soll. Gegebenenfalls erfolgt dann die Fehlermeldung ROM ERROR xxxx dd # mit Übergang zur entsprechenden Datenzelle. Wird Enter ohne Neueingabe gedrückt, so geht ZETBUG einfach zur nächsten Speicherzelle weiter. Mit einem „." kann man den Modify-Modus jederzeit wieder verlassen.

Beispiel:

```
# M 4A00   Enter
4A00   5B   # 77   Enter
4A01   43   # ED 53 20 40   Enter
4A05   09   # Enter
4A06   C9   # C8.   Enter
# T:   Enter
4A00   77   ED 53 20 40 09 C8...
#
```

Anmerkung:

Bei Verlassen des Modify-Modus wird zusätzlich das Argument 2 (bbbb) auf die zuletzt bearbeitete Adresse gesetzt. Auf diese Weise kann z. B. durch # T: der gerade modifizierte Bereich ausgedruckt werden. Bei allen Kommandos kann auch über den Zeilenrand hinaus geschrieben werden.

 # C aaaa bbbb cccc
(Copy memory)

Kopiert den Speicherbereich von aaaa mit der Länge cccc nach bbbb. Die beiden Speicherbereiche dürfen sich auch überlappen.

Beispiel: # C 4BC8 4BC9 3F
verschiebt einen 3F-langen Block um 1 nach oben.

 # K aaaa bbbb cc
(Kill memory contents)

Alle Adressen von aaaa bis bbbb werden mit der Konstanten cc geladen.

 # A aaaa bbbb c
(Arithmetic)

Addiert die beiden Hexzahlen aaaa und bbbb; subtrahiert bbbb von aaaa; errechnet das „Displacement" von aaaa nach bbbb, wobei c die Länge des Sprungbefehls in Bytes angibt; errechnet den dezimalen Wert der Hexzahl aaaa.

Beispiel:
A 447B 4472 2
DSPL:F5 SUM:88ED DIF:FFF7
DEC:17531
#

 # V aaaa bbbb cccc
(Verify memory)

Vergleicht die beiden Speicherbereiche von aaaa und bbbb der Länge cccc. Werden zwei ungleiche Bytes gefunden, so erfolgt die Meldung: xxxx bb yyyy d, wobei xxxx die Speicherzelle bb aus dem Block ab aaaa und yyyy die entsprechende Zelle dd aus dem Block ab bbbb ist. Durch Drücken der Enter-Taste wird mit dem Vergleich fortgefahren. Jede andere Taste führt zum Abbruch des Kommandos.

 # F aa bb cc dd ee ff ... nn
(Find data string)

Sucht im Speicher die Datenfolge aa bb cc ... nn, wobei beliebig viele Bytes angegeben werden können. Die Suche beginnt in der Zeropage (genau ab 00aa). Wird der Datenstring gefunden, so geht ZETBUG in den Modifymodus über (siehe M), indem er auf die Anfangsadresse des Strings zeigt. Ansonsten erfolgt die Fehlermeldung: NOT FOUND # –.

Der Befehl F versteht die :–Option auf besondere Weise:

F: aa bb cc dd ee ... nn
bedeutet, daß mit der Suche an der Stelle im Speicher fortgefahren werden

```
43B0 00 00 50 00 00 00 00 00 D5 FD E5 CD 2B 00 B7 28
43C0 FA FD E1 D1 C9 F5 D5 FD E5 FE 08 28 08 C4 33 00
43D0 FD E1 D1 F1 C9 2A 20 40 3E 23 2B 2B BE 3E 08 18
43E0 EC E3 7E 23 B7 28 05 CD C5 43 18 F6 E3 C9 F5 CD
43F0 E1 43 20 23 20 00 CD B8 43 CD C5 43 FE 0D 20 F6
4400 E5 2A 20 40 3E 23 ED A9 20 FC 23 23 23 22 93 43
4410 E1 F1 C9 1A FE 20 13 28 FA 1B AF 21 90 43 77 23
4420 77 23 77 1A 2B 2B D6 30 F8 FE 0A 38 08 D6 07 FE
4430 0A F8 FE 10 F0 13 34 23 ED 6F 23 ED 6F 18 E4 C5
4440 F5 CD 13 44 23 44 4D 6E 03 0A 67 C1 B5 78 C1 C9
4450 F5 1F 1F 1F 1F CD 59 44 F1 F5 E6 0F C6 30 FE 3A
4460 38 02 C6 07 CD C5 43 F1 C9 F5 7C CD 50 44 7D CD
4470 50 44 F1 C9 CD E1 43 57 48 41 54 3F 20 20 00 CD
4480 EE 43 21 C0 44 01 FF 0F AF 03 ED B1 20 E6 3E 09
4490 BE 20 F5 23 ED 5B 93 43 1A BE 20 EC 23 7E FE 0D
44A0 20 E6 23 E5 13 CD 3F 44 20 05 1A FE 3A 28 0F 22
44B0 95 43 CD 3F 44 22 97 43 CD 3F 44 22 99 43 E1 01
44C0 7F 44 C5 E9 00 09 20 0D C9 31 90 43 CD E1 43 0E
44D0 0D 0D 5A 45 54 42 55 47 20 20 20 5A 38 30 2D 4D
44E0 4F 4E 49 54 4F 52 20 20 20 56 31 2E 30 0D 0D 00
44F0 00 09 49 0D 21 90 43 54 5D 13 36 00 01 27 00 ED
4500 B0 3E 50 32 B2 43 31 90 43 CD E1 43 43 4C 52 2F
4510 52 53 41 0D 00 C3 7F 44 00 09 54 0D 2A 95 43 ED
4520 5B 97 43 E5 ED 52 E1 D0 CD 69 44 06 10 CD E1 43
4530 20 00 7E CD 50 44 23 10 F4 CD E1 43 0D 00 18 DF
4540 00 09 4D 0D 2A 95 43 CD 69 44 CD E1 43 20 20 00
4550 7E CD 50 44 CD EE 43 ED 5B 93 43 2B 23 E5 CD 3F
4560 44 28 17 7D E1 77 BE 28 F3 CD E1 43 52 4F 4D 20
4570 45 52 52 4F 52 20 20 00 18 CD 1A FE 20 28 E4 E1
4580 23 22 97 43 FE 2E C8 ED 5B 93 43 1A FE 20 28 E8
4590 2B 18 E5 2A 95 43 ED 5B 97 43 ED 4B 99 43 C9 00
45A0 09 4B 0D CD 93 45 71 E5 AF EB ED 52 44 4D E1 54
45B0 5D 13 ED B0 C9 00 09 43 0D CD 93 45 AF E5 ED 52
45C0 E1 30 EF 09 EB 09 EB 2B 1B ED B8 C9 00 09 41 0D
45D0 CD 93 45 E5 D5 09 EB ED 52 3E 00 38 09 BC 20 1B
45E0 CB 7C 20 17 18 08 2F BC 20 11 CB 7C 28 0D CD E1
45F0 43 44 53 50 4C 3A 00 7D CD 50 44 D1 E1 E5 E5 19
4600 CD E1 43 20 20 20 53 55 4D 3A 00 CD 69 44 E1 ED
4610 52 CD E1 43 20 20 20 44 49 46 3A 00 CD 69 44 C1
4620 21 00 00 5D 78 B1 28 0F 0B 7D 3C 27 6F 7C CE 00
4630 27 67 30 F0 1C 18 ED CD E1 43 20 20 20 44 45 43
4640 3A 00 7B E6 0F F6 30 CD C5 43 CD 69 44 CD E1 43
4650 0D 00 C9 00 09 56 0D CD 93 45 1A BE 20 08 0B 23
4660 13 78 B1 C8 18 F4 CD 11 47 7E CD 50 44 CD 14 47
4670 CD 14 47 EB CD 11 47 EB 1A CD 50 44 CD 4D 46 CD
4680 B8 43 FE 0D C0 18 D7 00 09 46 0D ED 4B 95 43 ED
4690 5B 93 43 13 13 03 CD 3F 44 0A BD 28 07 03 78 B1
46A0 28 18 18 F5 C5 CD 3F 44 03 0A BD 28 F8 1A FE 20
46B0 C1 28 DC ED 43 95 43 C3 44 45 CD E1 43 0E 20 4E
46C0 4F 54 20 46 4F 55 4E 44 0D 00 C9 CD E1 43 53 50
46D0 3A 00 18 3D CD E1 43 50 43 3A 00 18 34 CD E1 43
46E0 49 59 3A 00 18 2B CD E1 43 49 58 3A 00 18 22 CD
46F0 E1 43 48 4C 3A 00 18 19 CD E1 43 44 45 3A 00 18
4700 10 CD E1 43 42 43 3A 00 18 07 CD E1 43 41 46 3A
4710 00 CD 69 44 CD E1 43 20 20 20 20 00 C9 00 09 52
4720 0D FE 3A C2 FF 47 CD E1 43 42 50 3A 00 2A B3 43
4730 CD 11 47 CD E1 43 42 53 3A 00 21 B5 43 7E CD 50
4740 44 23 7E CD 50 44 23 7E CD 50 44 CD 4D 46 2A B1
4750 43 CD CB 46 2A AF 43 CD D4 46 2A AD 43 CD DD 46
4760 2A AB 43 CD E6 46 CD E1 43 0D 4D 41 49 4E 20 20
4770 00 2A A9 43 CD EF 46 2A A7 43 CD F8 46 2A A5 43
4780 CD 01 47 2A A3 43 CD 0A 47 CD E1 43 0D 45 58 58
4790 52 20 20 00 2A A1 43 CD EF 46 2A 9F 43 CD F8 46
47A0 2A 9D 43 CD 01 47 2A 9B 43 CD 0A 47 CD E1 43 0D
47B0 46 4C 41 47 53 3A 20 20 00 3A A3 43 CD D3 47 CD
47C0 E1 43 20 20 28 00 3A 9B 43 CD D3 47 CD E1 43 29
47D0 0D 00 C9 6F CB 7D C4 E1 43 53 00 CB 75 C4 E1 43
47E0 5A 00 CB 65 C4 E1 43 48 00 CB 55 C4 E1 43 50 00
47F0 CB 4D C4 E1 43 4E 00 CB 45 C4 E1 43 43 00 C9 2A
4800 93 43 23 23 7E 11 A3 43 FE 41 28 12 13 13 FE 42
4810 28 0C 13 13 FE 44 28 06 13 13 FE 48 20 2B 23 23
4820 7E FE 27 EB 20 05 11 08 00 ED 52 5E 23 56 EB CD
4830 69 44 CD EE 43 2A 93 43 EB E5 D5 CD 3F 44 EB E1
4840 7E FE 20 E1 C8 72 2B 73 C9 13 13 23 7E FE 58 28
4850 13 13 13 FE 59 28 0D 13 13 FE 43 28 07 13 13 FE
4860 50 C2 74 44 EB 18 C4 ED 73 90 43 31 AF 43 FD E5
4870 DD E5 E5 D5 C5 F5 D9 08 E5 D5 C5 F5 18 15 ED 73
4880 90 43 31 9B 43 F1 C1 D1 E1 D9 08 F1 C1 D1 E1 DD
4890 E1 FD E1 ED 7B 90 43 C9 CD 67 48 E1 ED 73 B1 43
48A0 31 90 43 CD E1 43 42 52 45 41 4B 20 49 4E 3A 00
48B0 2B 2B 2B 22 AF 43 CD 11 47 2A B3 43 11 B5 43 01
48C0 03 00 EB ED B0 31 90 43 C3 7F 44 00 09 42 0D 2A
48D0 95 43 22 B3 43 11 B5 43 01 03 00 ED B0 C9 00 09
48E0 45 0D 2A B3 43 36 CD 23 36 98 23 36 48 00 09 4A
48F0 0D 2A 95 43 22 AF 43 ED 7B B1 43 E5 C3 7E 48 00
4900 09 47 0D 2A AF 43 22 95 43 ED 5B B3 43 AF ED 52
4910 20 D0 18 DD 00 09 50 0D 2A 20 40 E5 21 8D 05 22
4920 1E 40 CD 1C 45 21 58 04 22 1E 40 E1 22 20 40 C9
4930 00 09 4C 0D 1A ED 5B 95 43 FE 2D 20 0A 21 00 00
4940 ED 52 EB ED 53 95 43 AF CD 12 02 CD 96 02 06 07
4950 CD 35 02 10 FB CD 35 02 FE 3C 20 2C CD 35 02 47
4960 CD 35 02 6F CD 35 02 67 85 4F ED 5B 95 43 19 CD
4970 35 02 77 81 4F 23 10 F7 CD 35 02 B9 00 00 CD 35
4980 02 FE 78 20 D3 C3 F8 01 CD E1 43 43 3F 0D 00
4990 18 F3 00 09 44 0D 13 1A FE 20 28 FA AF CD 12 02
49A0 CD 87 02 3E 55 CD 64 02 06 06 1A CD 64 02 13 10
49B0 F9 ED 5B 95 43 2A 97 43 23 ED 52 28 2F 01 00 01
49C0 E5 ED 42 E1 38 02 2E 00 45 3E 3C CD 64 02 78 CD
49D0 64 02 7B CD 64 02 7A CD 64 02 83 4F 1A 81 4F 1A
49E0 CD 64 02 13 10 F6 79 CD 64 02 18 C9 3E 78 CD 64
49F0 02 3A 99 43 CD 64 02 3A 9A 43 CD 64 02 C3 F8 01
```

Bild 1. ZETBUG umfaßt etwa 1,6 KByte und wurde aus Platzgründen hier nur hexadezimal aufgelistet. Zur Eingabe kann man das Basic-Programm in Bild 2 verwenden

soll, an der sie zuletzt unterbrochen wurde (bzw. ab der Adresse, die zuletzt als Argument 1 angegeben wurde). Jedesmal, wenn der Datenstring gefunden wird, legt ZETBUG die Startadresse des Strings in der Memory-map ab, und zwar in die Speicherzellen, in denen das Argument 1 steht.

Auf diese Weise kann auf jeden String im Speicher zugegriffen werden, auch wenn dieser an mehreren Stellen gleichzeitig vorkommt.

 # D aaaa bbbb cccc name
(dump data to cassette)

Hiermit lassen sich Datenblöcke (files) auf Kassette aufzeichnen; aaaa ist die Anfangsadresse, bbbb ist die Endadresse, cccc ist der Entrypoint (siehe auch LEVEL II-System-Kommando); name ist der Filename (maximal 6 Zeichen). Aufzeichnungsformat (ZETBUG-Tapes können direkt mit dem Level-II-System-Kommando gelesen werden):

- 255 Byte Headercode
- Synchronisationsbyte (A5)
- filename Headercode (55)
- 6 Byte langer file-name
- Datenblöcke:
  - data Headercode (3C)
  - Anzahl der Datenbytes in diesem Block (00-FF) (00=256x)
  - Blockstartadresse
  - (LSB/MSB)
  - Daten
  - Checksumbyte
- Entrypoint Headercode (78)
- Entrypoint (LSB/MSB)

 # L aaaa±
(Load data from cassette into memory)

Liest einen File von der Kassette in den Speicher. aaaa± gibt das Displacement an, um das der File versetzt geladen werden soll (kann entfallen).

L 1200+ bedeutet, daß der File auf eine um 1200 höhere Adresse als ursprünglich geladen werden soll.

 # P aaaa bbbb
(Print data)

Druckt den Speicherbereich von aaaa bis bbbb auf dem Lineprinter aus. (Das Format entspricht dem des T-Kommandos.)

 # E aaaa
(Execute machine program)

Führt das Maschinenprogramm ab der Startadresse aaaa aus. Zuerst wird der Breakpoint (siehe B) auf die durch das B-Kommando festgesetzte Adresse gelegt, dann werden sämtliche CPU-Register mit den Daten aus der RSA (siehe auch I) geladen und zur Programmadresse aaaa gesprungen.

 # J aaaa
(Jump machine program)

Wie das E-Kommando, jedoch wird kein Breakpoint gesetzt.

 # G
(Go on / continue execution at PC)

Durch Eingabe von G wird die Programmausführung dort fortgesetzt, wo sie zuletzt (z. B. durch einen Breakpoint) unterbrochen wurde. Nachdem die CPU-Register geladen wurden, wird geprüft, ob die PC-Adresse (PC = Programcounter der Z-80-CPU) auf den Breakpoint zeigt (das nämlich ist genau dann der Fall, wenn der Programmstop durch eben diesen Breakpoint verursacht wurde). Ist dem so, dann wird direkt zur PC-Adresse gesprungen, ansonsten wird zuerst der Breakpoint gesetzt und dann zur PC-Adresse gesprungen.

 # B aaaa
(Breakpoint)

Dies ist wohl einer der hilfreichsten Befehle zum Austesten von Maschinenprogrammen. Er bietet die Möglichkeit, an jeder Stelle im Programm einen Software-Haltepunkt (Breakpoint) zu setzen, ohne das Programm verändern zu müssen. Prinzip: An die für den Break ausgewählte Stelle, die immer die Adresse des ersten (!) Bytes eines Maschinenbefehls sein muß, wird eine 3 Byte lange „CALL BREAK"-Sequenz (CD 98 48) gelegt. Gelangt das Programm bei der Ausführung an diese Stelle, so springt es zu der Breakpointbehandlungsroutine BREAK in den ZETBUG-Monitor. Diese rettet nun die Inhalte der CPU-Register in die RSA, ersetzt im Speicher die Breakpointsequenz durch die ursprüngliche Opcodefolge, druckt die Meldung: BREAK IN aaaa #– und gibt die Kontrolle wieder zurück an den Benutzer (man kann nun z. B. das Kommando R: eingeben, siehe R).

Anzumerken ist noch, daß die Breakpointsequenz wirklich nur dann im Speicher steht, wenn gerade ein durch E oder G gestartetes Maschinenprogramm von der CPU abgearbeitet wird. Dem Programm merkt man also nichts an, während man z. B. mit dem M-Kommando arbeitet.

 # R
(Register display / modify)

Mit diesem Befehl läßt sich direkt auf die Register Save Area (RSA) zugreifen, ohne die Speicheradressen der einzelnen CPU-Register wissen zu müssen. Die Eingabe # R/XX bewirkt, daß der Inhalt des Register(paares) XX angezeigt wird und im Anschluß an das Promptsymbol die neuen Daten (ähnlich wie im Modify-Modus) eingegeben werden können. Das Kommando # R: druckt die gesamte RSA einschließlich der Breakpointsequenz und der Opcodefolge mit Benennung auf dem Bildschirm aus. Außerdem werden die Flagregister interpretiert und die gesetzten Flags abgekürzt angegeben (F' in Klammern). Auf diese Weise unterstützt der R-Befehl wirkungsvoll die Kommandos E, J, G und B. Beispiel:

```
# R:
BP:534D  BS:212040
SP:4FFE  PC:534D  IY:6CDD  IX:3000
MAIN  HL:21DB  DE:8F53  BC:4433
  AF:3EAA
EXXR  HL:1277  DE:9CCC  BC:E841
  AF:5503
FLAGS:  SN  (NC)
#
#  R/DE'
9CCC # 8888
#  R/DE'
8888
#
```

```
100 DEFINT A-Z
110 PRINT:PRINT"ADRESSE:"; :GOSUB1000:Z=D*256:GOSUB1000:Z=Z+D
200 PRINT:D=Z/256:GOSUB1200:D=Z-D*256:GOSUB1200:PRINT CHR$(32);
210 D=PEEK(Z+512):GOSUB1200:PRINT CHR$(32); :GOSUB1000:IF X=33 THEN200
220 IF X=23 THEN280 ELSE IF X=22 THEN100 ELSE IF X=29 END
230 POKE(Z+512),D
280 Z=Z+1:GOTO 200
1000 GOSUB1020:IFX>16 RETURN ELSE D=X*16:GOSUB1020:D=D+X:RETURN
1020 X$=INKEY$:IF X$="" THEN1020 ELSE X=ASC(X$)
1030 PRINT CHR$(X); :IF X<64 THEN X=X-48:RETURN ELSE X=X-55:RETURN
1200 A=D/16:IF A>9 THEN A=A+55 ELSE A=A+48
1210 B=D/16:B=D-B*16:IF B>9 THEN B=B+55 ELSE B=B+48
1220 PRINT CHR$(A); :PRINT CHR$(B); :RETURN
```

Bild 2. Basic-Programm zur Eingabe hexadezimaler Maschinenprogramme. Zeile 100 definiert die verwendeten Variablen als speicherplatzsparende Ganzzahlen. Der INKEY-Befehl entspricht beim PET der GET-Anweisung. ELSE IF X = 29 END müßte beim PET in Zeile 225 als IF X = 29 THEN END stehen

 # aaaa bbbb cccc

Neben den 16 beschriebenen ZETBUG-Kommandos gibt es noch diesen Blank- oder Leerbefehl. Er dient lediglich zum Abspeichern der Argumente, wenn im weiteren Verlauf mit der :–Option gearbeitet werden soll oder die Argumente aus einem anderen Grunde modifiziert werden müssen. Ohne Angabe von Argumenten (einfaches Drükken von ENTER) werden der Bildschirm um eine Zeile nach oben „gescrolled" und die Argumente auf 0000 gesetzt. Anwendung:

```
#     4000
# F:00  09  52  0D
471D  00 #
#
```

Hier bewirkt der Blankbefehl, daß die Datensuche erst ab der Adresse 4000 beginnen soll.

Erweiterung des Befehlssatzes

ZETBUG ist erweiterbar, denn er enthält keine Befehlstabelle, in der jedes Kommando katalogisiert ist, sondern „sucht" sich den vom Benutzer definierten Befehl im Speicher. Jedes Kommando hat an der Stelle, an der es von der ZETBUG-Hauptroutine angesprungen werden soll, seine eigene Codemarke. Diese hat die Form: 00 09 XX 0D ... routine ... C9, wobei XX der ASCII-Code des ZETBUG-Befehls ist. So lautet z. B. der Code für den R-Befehl: 00 09 52 0D.

Wird der Code in einem 4 KByte großen Bereich (ab 44C0) nicht gefunden, so druckt ZETBUG die Meldung WHAT? und verlangt eine neue Befehlszeile. Diese Methode erlaubt damit z. B. das unproblematische Hinzufügen zusätzlicher Kommandos, ohne ZETBUG abändern zu müssen. Allerdings muß beachtet werden, daß jede Kommandoroutine mit einem Z-80-Returnbefehl zu ZETBUG zurückkehren sollte (Achtung auf den Stack!).

ZETBUG auf anderen Systemen

Um ZETBUG auch auf anderen Systemen zu implementieren, muß man einige Veränderungen vornehmen. TRS-80/Level-2-spezifisch sind die INCH-Routine, die OUTCH-Routine und die Kommandos L,D,P. Zu beachten ist außerdem, daß die INLINE-Routine die Cursorposition (Level 2:4020) benötigt, um die Eingabezeile und deren Zeilenanfang zu errechnen. Die *Tabelle* zeigt die Adressen von ZETBUG.

Eingeben von ZETBUG in den Speicher

Um ZETBUG das erste Mal in den Speicher zu laden, kommt man leider nicht umhin, die 1,6 K Maschinencode von Hand einzutippen. Das geschieht mit dem BASIC-Eingabeprogramm (*Bild 2*). Hat man diese Fleißleistung dann glücklich vollbracht, so empfiehlt sich, alles nochmals nachzukontrollieren. Dann kann man ab 4C00 folgendes Maschinenprogramm eingeben:

21 B8 45 11 B8 43 01 FF 07 ED B0 C3 C9 44

Starten (im Level 2 BASIC):

```
>SYSTEM
*? / 19968   Enter
```

Sodann meldet sich ZETBUG. Jetzt als erstes auf Band sichern: # D 43B8 49FF 44C9 ZETBUG (besser mehrmals).

Tabelle: ZETBUG – Adressen und Unterprogramme (UP)

Memory-map	4020+4021	Cursorposition (Level II)
	ca. 4380 – 438F	ZETBUG Stack
	4390 – 4392	data
	4393+4394	start of input line (siehe INLINE)
	4395+4396	Argument 1
	4397+4398	Argument 2
	4399+439A	Argument 3
RSA	ab 439B	in der Reihenfolge:
Register		AF'/BC'/DE'/HL'/AF/BC/DE
Save		HL/IX/IY/PC/SP
Area	43B3+43B4	Breakpointadresse
	43B5–43B7	Opcodesequenz (3 Bytes)
ENTRY	44C9	ZETBUG-Entrypoint. Wird angesprungen,
(=17609 dez.)		wenn ZETBUG aufgerufen werden soll (z. B. nach dem Laden von der Kassette).
RETURN	48B9	Adresse, die ein Benutzerprogramm anspringen
(C3 B9 48)		muß, wenn es zu ZETBUG zurückkehren will. Keines der CPU-Register wird gerettet!
BREAK	4898	Breakpointbehandlungsroutine (siehe B). Kann
		außerdem als Rücksprungadresse von einem Benutzerprogramm verwendet werden, wenn
(CD 98 48)		die CPU-Register in die RSA gerettet werden sollen.
LOAD	487E	Unterprogramm schreibt die RSA in die CPU zurück (außer SP- und PC-Register!)
SAVE	4867	UP rettet die CPU-Register (außer SP und PC) in die RSA.
INCH	43B8	UP liest ein ASCII-Zeichen von der Tastatur. Character in accu.
OUTCH	43C5	UP setzt ein Zeichen auf den Bildschirm und bewegt den Cursor (character in accu).
PRINT	43E1	Aufruf: CD E1 43 aa bb cc dd ... nn 00. UP druckt den ASCII-string aa bb ... nn auf den Bildschirm und bewegt den Cursor.
INLINE	43EE	UP liest eine Zeile vom Bildschirm (wickelt alle notwendigen INCH/OUTCH-Routinen ab) und errechnet den Zeilenanfang (start of line in 4393+4394). Der Zeilenanfang ist immer zwei Zeichen nach dem Promptsymbol.
INHEX	443F	UP liest eine 4-digit-Hexzahl vom Schirm in das HL-Register. Der Bufferpointer ist das DE-Register (wird aut. nachkorrigiert).
OUTHEX	4450	UP druckt den Accu im Hexformat aus.
OUTHL	4469	wie OUTHEX, jedoch HL-Register.

ZETBUG belegt den Speicherbereich von 43B8 bis 49FF.
Das Benutzerprogramm kann ab 4A00 stehen.
Der Stackpointer ist normalerweise auf 5000hex gesetzt, kann aber verändert werden (durch R/SP, siehe R).

Michael Klein

Wie funktioniert der direkte Speicherzugriff (DMA)?

Der direkte Speicherzugriff stellt eine Möglichkeit dar, ohne Belastung der CPU mit hoher Geschwindigkeit Daten zu transferieren und wird daher oft z. B. in Floppy-Disk-Systemen angewendet. Der Autor gewann im FUNKSCHAU-Preis 1979 mit nachfolgendem Beitrag 500 DM.

Motivation für den DMA-Einsatz

Immer mehr werden Mikrorechnersysteme in Technologie und Architektur mit wesentlich größeren Systemen vergleichbar. Zunehmend werden auch jene Techniken bei Mikrorechnersystemen eingesetzt, die noch Ende der 60er Jahre fast ausschließlich den größeren Rechnersystemen vorbehalten waren. Ein Beispiel dafür ist das DMA-Konzept, der direkte Speicherzugriff (DMA engl. „direct memory access").

Sinn einer DMA ist es, unter Umgehung der Zentraleinheit (CPU) einen schnellen Datentransport zwischen Speicherbereichen und Peripheriebereichen zu ermöglichen. Dies ist in der Regel immer dann besonders sinnvoll, wenn große Datenmengen mit hoher Geschwindigkeit übertragen werden müssen. Beim üblichen Buskonzept sind Speicher, Zentraleinheit und Peripheriebausteine über drei gemeinsame Sätze von Busleitungen gekoppelt: Datenbus, Adreßbus und Steuerleitungen. *(Bild 1)*.

Allein die CPU steuert sämtliche Vorgänge im Bussystem.Um z. B. eine Hexadezimalzahl aus der Speicherzelle mit der Adresse 50 in die Speicherzelle 51 zu übertragen, muß die Zentraleinheit aktiviert werden. Zwei Befehle sind abzuarbeiten: – Lade den Inhalt der Speicherzelle 50 in ein CPU-Register – Lade den Inhalt des CPU-Registers in die Speicherzelle 51. In der Regel reicht die Verarbeitungsgeschwindigkeit der CPU für solche Umspeicheraufgaben aus. Wenn man jedoch mit sehr großen Datenmengen arbeiten muß, erweist es sich als sinnvoll, daß man – ohne den Programmablauf in der CPU merklich zu stören oder zu verlangsamen – mittels DMA Daten zwischen zwei Funktionsblöcken des Rechners austauschen kann. *Bild 2* zeigt die Stellung des DMA-Bausteines (hier auch etwas salopp einfach als „DMA" bezeichnet) innerhalb eines Systemes mit klassischem Buskonzept. Die DMA muß, um Daten übertragen zu können, wie die CPU Adreßbus, Datenbus und Steuerleitungen bedienen können.

Damit sich DMA und CPU nicht bei der Arbeit gegenseitig stören, müssen zwischen den beiden Bausteinen Steuersignale ausgetauscht werden. Während die DMA aktiv ist, werden die Bustreiberausgänge der CPU in den hochohmigen dritten Zustand (Tristate) gebracht. Man spricht davon, daß der CPU-Bus „gefloatet" wird. Wenn die CPU aktiv ist, floaten die DMA-Ausgänge.

Im folgenden Abschnitt soll dargestellt werden, was eine DMA leisten kann, wie man sie einsetzt und programmiert. Stellvertretend für die große Zahl von DMA-Bausteinen wurde hier die Z-80-DMA ausgewählt, die der Autor seit einiger Zeit an einem Nascom-1-Mikrorechnersystem betreibt. Die folgenden Ausführungen treffen aber auch auf die meisten anderen DMA-Bausteine zumindest im Grundsätzlichen zu.

Betrieb der DMA

Funktionstypen

Drei Typen von Operationen, sog. „Funktionstypen", kennzeichnen die Z-80-DMA:

1. Schnelle *Übertragung* von Daten zwischen zwei Funktionsblöcken. Ein Funktionsblock ist z. B. der Arbeitsspeicher oder ein Peripherie-

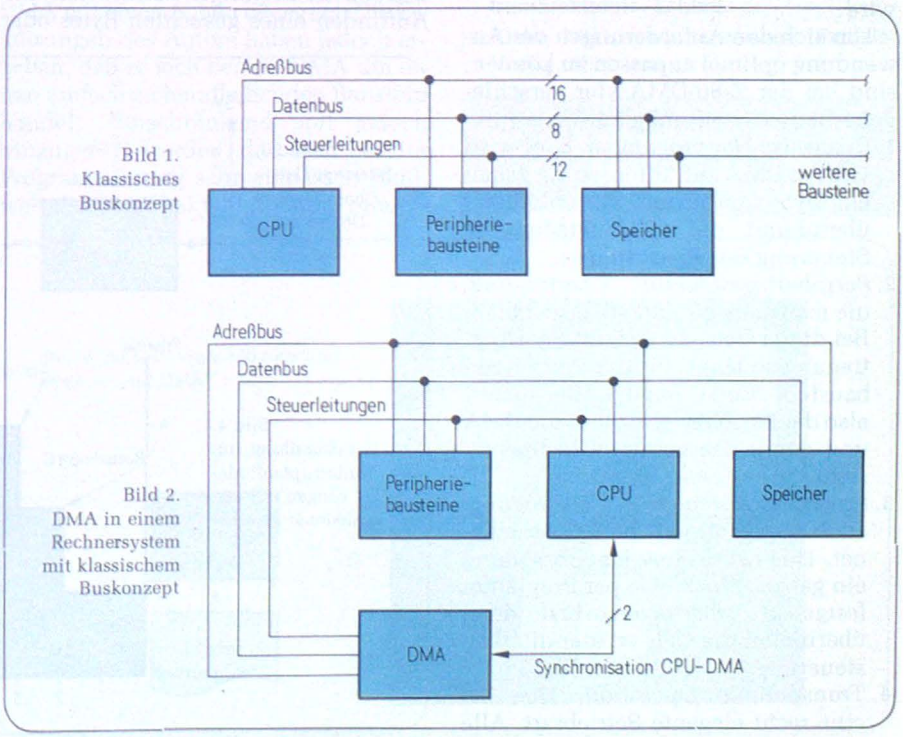


Bild 1. Klassisches Buskonzept

Bild 2. DMA in einem Rechnersystem mit klassischem Buskonzept

baustein. In dieser Betriebsart werden etwa 1,2 Megabyte pro Sekunde übertragen.

2. Schnelle *Suche* nach einem Byte. In dieser Betriebsart durchsucht die DMA einen Speicherblock nach einem bestimmten Byte. Wenn das Byte gefunden wurde, meldet die DMA dies an die CPU z. B. über einen Interrupt.
3. In der Betriebsart *Suche und Übertragung* wird ein bestimmtes Byte in einem Datenblock gesucht und wenn es gefunden ist, an einen Funktionsblock übertragen. Man kann die DMA auch so programmieren, daß sie aus einem Speicherblock genau so lange Daten überträgt, bis sie ein bestimmtes Byte gefunden hat, z. B. ein EOT-Zeichen (Abk. engl. „end of text"). Dann wird die Übertragung gestoppt.

Betriebsarten

Zu Beginn unserer Darstellung wurde bereits erwähnt, daß beim direkten Speicherzugriff der DMA-Baustein die Bus-Steuerung übernimmt. Während die DMA aktiv ist, kann daher die CPU nicht arbeiten.

Je nach Anwendung mag es wünschenswert erscheinen, eher wenige länger andauernde DMA-Operationen zuzulassen oder lieber zahlreiche kurze Operationen. Bei Systemen mit dynamischen Speichern muß bedacht werden, daß die Speicher nicht aufgefrischt werden können, während die DMA arbeitet. Man muß dann dafür sorgen, daß die CPU nie länger als 2 ms angehalten wird.

Um sich den Anforderungen der Anwendung optimal anpassen zu können, sind bei der Z-80-DMA vier verschiedene Betriebsarten möglich:

1. *Byteweise Übertragung*, d. h. es wird von der DMA auf Anforderung genau ein Byte übertragen. Anschließend übernimmt die CPU wieder die Steuerung des Bussystemes.
2. *Peripheriegesteuerte Übertragung*, die man auch als „burst" bezeichnet. Bei dieser Betriebsart läuft die Übertragung so lange, bis der Peripheriebaustein „Ende" meldet. Hier steuert also der Peripheriebaustein die DMA und damit das gesamte Rechnersystem.
3. *Programmgesteuerte Übertragung*, auch als „continous mode" bezeichnet. Hier wird ohne Unterbrechung ein ganzer Block, wie per Programm festgelegt, übertragen. Erst dann übernimmt die CPU wieder die Bussteuerung.
4. *Transparente Operation*. Dies ist eine recht elegante Betriebsart. Alle DMA-Operationen laufen bei dieser Betriebsart während der Speicherauffrischzyklen ab, die die Z-80-CPU automatisch erzeugt, um die dynamischen RAMs aufzufrischen. Dabei läuft der direkte Speicherzugriff ohne Zeitverlust ab!

Steuerung der DMA; Zeitverhalten

Es soll hier nicht in Details auf die Programmierung der Z-80-DMA eingegangen werden, dies ist Aufgabe des entsprechenden Handbuches [1]. Es sollen jedoch einige Besonderheiten der Z-80-DMA herausgestellt werden, die bemerkenswert erscheinen.

Eine sehr interessante Eigenschaft der Z-80-DMA ist die Möglichkeit, das Zeitverhalten beim Speicherzugriff zu programmieren. Die Z-80-CPU kann mit WAIT-Zyklen auch Speicher mit längerer Zugriffszeit als 450 ns bedienen. Die DMA verhält sich beim Lesen und Beschreiben von Speicherinhalten genauso wie die CPU. Sie ist allerdings in der Lage, für ihren Speicherzugriff 1...4 WAIT-Zyklen einzubauen. D. h. sie kann 1...4 Zyklen warten, bevor sie auf den Speicherinhalt zugreift. Damit ist es möglich, mit Speichern ganz verschiedener Zugriffszeiten zu arbeiten, ohne daß die übliche Zusatz-Hardware (Wait-Flipflop) erforderlich wird.

Wie alle interruptfähigen Z-80-Bausteine bietet auch die Z-80-DMA die Möglichkeit zum Vektorinterrupt. Per Programm kann festgelegt werden, unter welchen Bedingungen eine Unterbrechungsanforderung gesendet werden soll. Bedingungen können z. B. das Auffinden eines gesuchten Bytes oder das Ende einer Operation sein. In einem Z-80-System können mehrere DMA eingebaut werden, die Interrupts anfordern können. Deshalb ist ähnlich wie bei den Z-80-Parallelschnittstellen auch bei den DMA-Bausteinen eine Prioritätskaskadierung vorgesehen. *Bild 3* zeigt das Prinzip.

Mit einem „1"-Signal wird einem DMA-Baustein an seinem IEI („Interrupt enable in")-Eingang angezeigt, daß er einen Interrupt anfordern darf. Der Baustein A soll die höchste Priorität haben. Deshalb wurde sein IEI-Eingang auf „1" festgelegt. Er darf zu jedem Zeitpunkt die niederer priorisierten Kanäle bei der Interruptbehandlung unterbrechen. Dies geschieht mit dem IEO-Ausgang („Interrupt enable out"). Mit dem IEO-Ausgang zeigt der höher priorisierte Baustein allen rechts von ihm liegenden Bausteinen an, daß sie keinen Interrupt anfordern können. Dabei ist das IEO-Signal auf „0". Dieses Prinzip der Prioritätsregelung nennt man „daisy-chain"-Prioritätssteuerung.

Bild 4 zeigt, wie eine Folge von Interruptbehandlungen zeitlich verschachtelt sein kann, damit die Prioritätsregelungen eingehalten werden.

Kommunikation zwischen CPU und DMA

Damit keine Zugriffskonflikte auftreten, muß die Übergabe der Bussteuerfunktion zwischen DMA und CPU synchronisiert werden. Es gibt dazu zwei Steuerleitungen: $\overline{\text{BUSRQ}}$, $\overline{\text{BUSAK}}$. Mit dem Ausgang $\overline{\text{BUSRQ}}$ auf „0" zeigt die DMA der CPU an, daß sie die Kontrolle

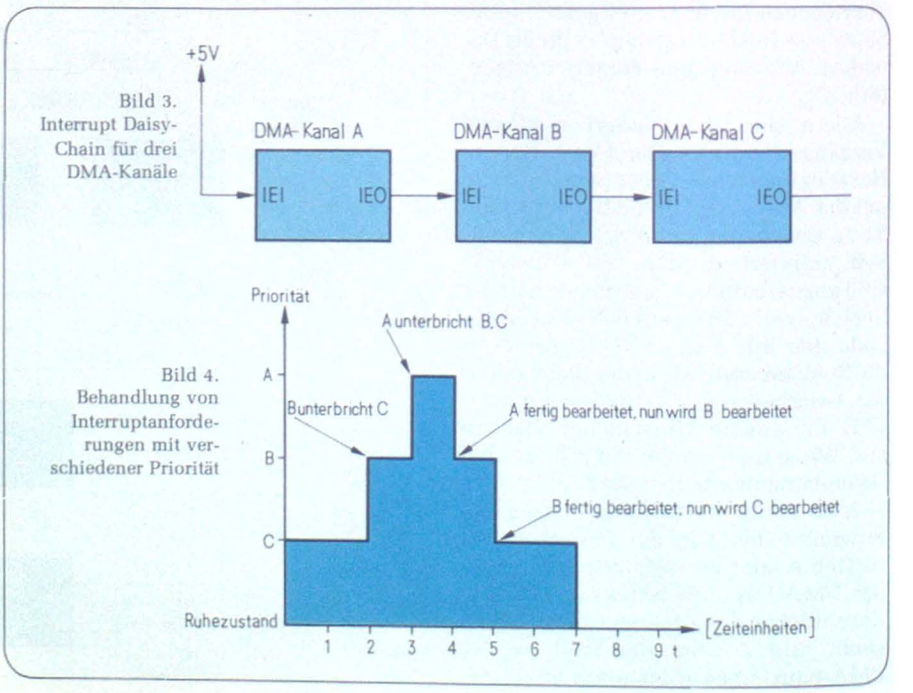


Bild 3. Interrupt Daisy-Chain für drei DMA-Kanäle

Bild 4. Behandlung von Interruptanforderungen mit verschiedener Priorität

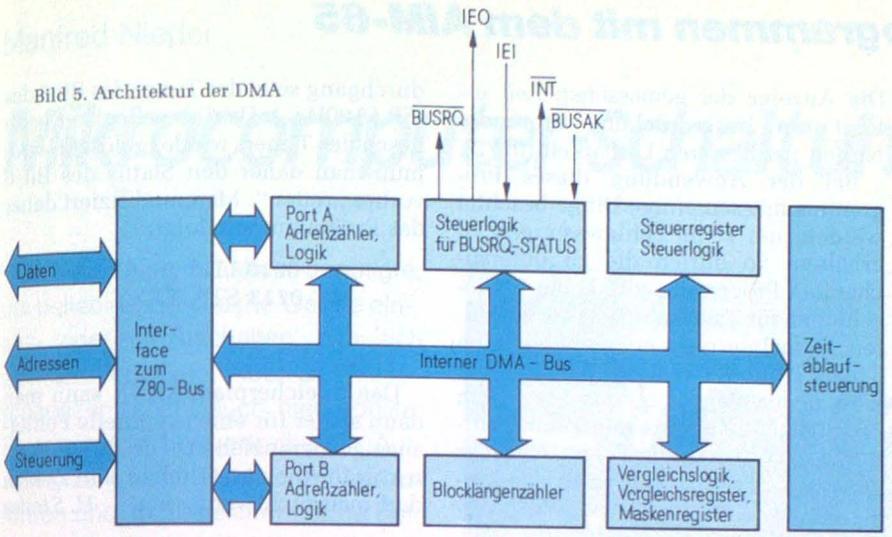


Bild 5. Architektur der DMA

über die Busleitungen wünscht. Wenn die CPU bereit ist, die Steuerung zu übergeben, legt sie als Quittung für den BUSRQ (engl. „bus request") den BUSAK-Ausgang (engl. „bus acknowledge") auf „0". Nun erst kann die DMA die Steuerung übernehmen. Wenn mehrere DMA in einem Z-80-System arbeiten, gibt es wieder eine „daisy chain", die bestimmt, welche DMA zuerst bedient wird.

Architektur und Programmierung der Z-80-DMA

Architektur

Obwohl hier nicht alle Möglichkeiten der DMA erläutert werden, geht aus dem schon Gesagten hervor, daß der DMA-Baustein über relativ viel „Intelligenz" verfügen muß. In der Tat ist die DMA nicht viel weniger komplex als die CPU. Die Architektur der DMA gibt *Bild 5* wieder.

Einen DMA-Kanal „sieht" der Prozessor im wesentlichen als zwei „Ports", also Ein-/Ausgabekanäle. Die DMA führt den Datenaustausch zwischen den Adreßbereichen durch, die durch die internen Portregister festgelegt sind. Jeder Port hat eine Anzahl Register, mit denen die Arbeitsweise der DMA festgelegt wird, sowie einen Satz von Speicherzellen (gemeinsam), deren Inhalt den Zustand des Systemes anzeigt (Statusregister). Ferner erkennt man noch den Blocklängenzähler, der angibt, wieviele Byte bereits übertragen wurden. Mit der Vergleichslogik und dem Maskenregister werden die Suchoperationen gesteuert.

Programmierung der DMA

Einige Steuerworte müssen der DMA übergeben werden. Sie geben an:
- Funktionstyp
- Betriebsart
- Anzahl der WAIT-Zyklen beim Speicherzugriff
- Interruptverhalten
- Interruptvektor
- zu suchendes Byte
- Maske für das zu suchende Byte

Man kann alle diese Steuerworte in linearer Reihenfolge im Speicher ablegen und der DMA mit dem Z-80-Befehl „OTIR" übergeben. Adreßregister und Blocklängenzähler sind gepufferte Register. D.h. es können schon für die nächste Operation Adreßregister und Blocklängenregister geladen werden, auch wenn die gerade laufende Operation noch nicht abgeschlossen ist.

Praktischer Einsatz der DMA

Noch immer beobachtet man, daß mancher Mikrorechneranwender der DMA mit etwas Scheu begegnet. Die Erfahrungen des Autors haben jedoch ergeben, daß es sich bei der DMA um einen einfach zu handhabenden Baustein handelt. Einschränkend soll jedoch hinzugefügt werden, daß man bei der Programmierung sehr aufpassen muß, weil die DMA bei fehlerhaftem Steuerwortsatz möglicherweise Speicherinhalte zerstören kann.

Mit recht gutem Erfolg wird die DMA in einem Nascom-1-Mikrorechnersystem eingesetzt. Es ist ihre Aufgabe, Listings von Basic-Programmen Zeile für Zeile an einen schnellen Nadeldrucker zu übergeben. Dabei wird die DMA in der Betriebsart „Suche und Übertragung" betrieben. Der DMA wird die Anfangsadresse des BASIC-Textes mitgeteilt. Außerdem wird ihr mitgeteilt, daß sie so lange den jeweils nächstfolgenden Speicherinhalt an die Parallelschnittstelle ausgeben soll, bis sie das Byte „1F" findet, das eine Textzeile abschließt. Nun wird eine Zeile über Parallelschnittstelle an den Nadeldrucker übergeben, die der Drucker speichern kann. Nachdem die Zeile gedruckt ist, meldet der Drucker über Interrupt, daß er fertig ist. Ein kleines Interruptserviceprogramm gibt der DMA die Anfangsadresse der nächsten Zeile und startet die DMA. Alles dies geschieht, während das Basic-Programm läuft, das gerade aufgelistet wird!

Bild 6 deutet die Konfiguration an. Die eingezeichnete „Hardware zur Quittungsübergabe" ist erforderlich, um die Datenübernahmevorschrift des Druckers einzuhalten.

Inzwischen hat sich die DMA beim Verfasser nicht nur bei der Druckausgabe, sondern auch noch bei anderen Anwendungen (Rechnerkopplung via DMA) recht gut bewährt.

Literaturangaben

[1] Z-80-DMA Produkt-Spezifikation. Kontron-Elektronik, Eching
[2] Klein, Mikrocomputersysteme. Franzis-Verlag, München

Stichworte zum Inhalt

Controller, Mikrocomputer, Z-80, Adressenbus, Datenbus

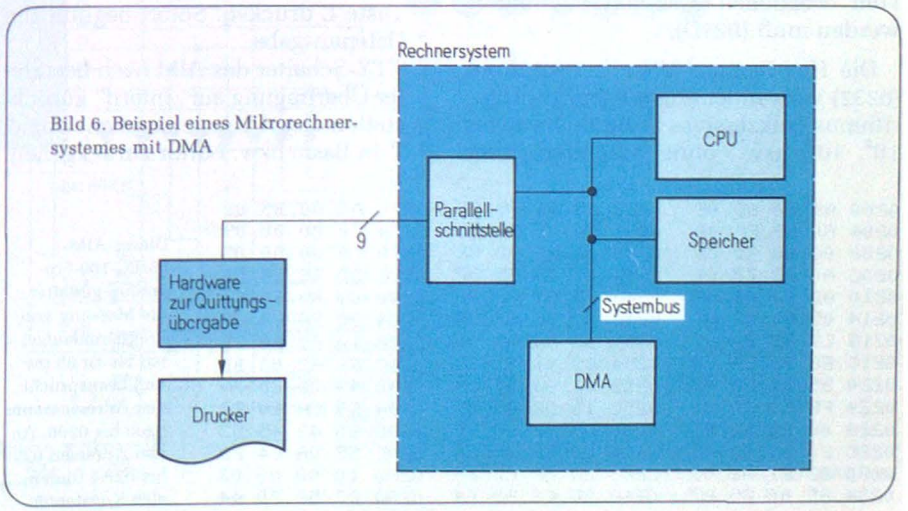


Bild 6. Beispiel eines Mikrorechnersystemes mit DMA

Zeitmessung von Programmen mit dem AIM-65

Bei der Erstellung von Maschinenprogrammen mit zeitkritischen Unterroutinen (wie z.B. Interruptroutinen, Online-Datenverarbeitung vom Tape) ist es oft erforderlich, den Zeitbedarf jeweils exakt zu bestimmen. Das ist zwar grundsätzlich über die Auszählung der Befehls-Zykluszeiten möglich, aber besonders bei Schleifen sehr mühsam und fehlerbehaftet.

Hier bietet sich der AIM selbst mit dem „glatten" Systemtakt von 1 µsec und seinen Timern geradezu als Zeitmeßgerät an. Das vorliegende Programm verwendet den Timer 1 im User-VIA für Messungen im Bereich von 1 bis max. 65524 µs mit einer Auflösung von 1 µs. Besonderer Wert wurde auf eine bequeme Handhabung und eine eindeutige, dezimale Zeitanzeige gelegt.

Das Programm arbeitet wie folgt: Beim Start (0200) wird der Timer mit $FFFF_{hex}$ geladen und gestartet. Danach erfolgt der Einsprung mit einem indirekten JMP über den Vektor in 0170/0171 in das zu testende Programm. An das Ende dieses Programms muß vorher die JMP-Anweisung zur Zeitmeßroutine angefügt werden, hier 0210.

Jetzt erfolgt die Auswertung des Timer-Inhaltes. Hier müssen zwei Korrekturen vorgenommen werden: Timer-High und Timer-Low können nur nacheinander ausgelesen werden. Hier kann es also passieren, daß zwischen den beiden Lesevorgängen noch ein Übertrag erfolgt. Wenn der invertierte Wert von T1-Low ≧FD ist, muß entsprechend der Wert in T1-High korrigiert werden (0218).

Durch die JMP-Befehle und den ersten Lesebefehl (T1-Low) entsteht eine Eigenverzögerung von 11 Zyklen, die vom erhaltenen Meßwert subtrahiert werden muß (021D).

Die Hex-Dezimal-Wandlungsroutine (0232) verwendet einen Primitiv-Algorithmus (sukzessives Subtrahieren von 10^4, 10^3, usw.) ohne Besonderheiten. Die Anzeige der gemessenen Zeit erfolgt mit Unterdrückung führender Nullen zur besseren Lesbarkeit (0272).

Bei der Anwendung dieses Programms müssen einige Dinge beachtet werden, um keine Fehlmessungen zu erhalten. So dürfen die zu untersuchenden Programme z.B. keine Warteschleifen für Tastenabfrage o. ä. enthalten, keine Interrupts erwarten oder den Timer 1 mit verwenden. Weiter sollte man überschlagen, ob das Programm evtl. länger als 65 ms sein kann. Fehlmessungen wären die Folge. Wer hier ganz sicher gehen will, kann das Programm etwas modifizieren: Beim Nulldurchgang setzt der Timer das Bit 6 des IFR ($A00D_{hex}$). Da dieses Register beim Lesen des Timers wieder gelöscht wird, muß man daher den Status des Bit 6 vorher „retten". Man modifiziert daher das Programm wie folgt:

```
0210 LDA A00D
0213 STA XXXX
      .
      .
```

Den Speicherplatz XXXX kann man dann später für eine eventuelle Fehleranzeige heranziehen. Die fällige Korrektur für die zusätzlichen acht Zyklen darf man nicht vergessen. *H. Steder*

```
0200 A9 40 8D 0E   0238 A5 01 85 08   0270 A0 00 B5 02
0204 A0 A9 FF 8D   023C A0 00 38 A5   0274 C0 00 D0 09
0208 06 A0 8D 05   0240 08 FD 9D 02   0278 C9 00 D0 05
020C A0 6C 70 01   0244 85 08 A5 07   027C 20 3E E8 D0
0210 AD 04 A0 AE   0248 FD A1 02 90   0280 06 09 30 C8
0214 05 A0 49 FF   024C 0C 85 07 C8   0284 20 7A E9 CA
0218 C9 FD 90 01   0250 85 00 A5 08   0288 10 E8 20 3E
021C E8 38 E9 0B   0254 85 01 4C 3E   028C E8 A2 03 BD
0220 85 01 8A 49   0258 02 94 03 CA   0290 99 02 20 7A
0224 FF E9 00 85   025C 10 D6 A5 01   0294 E9 CA 10 F7
0228 00 20 32 02   0260 85 02 60 20   0298 60 43 45 53
022C 20 63 02 4C   0264 3B E8 A0 50   029C 55 0A 64 E8
0230 82 E1 A2 03   0268 20 AF E7 20   02A0 10 00 00 03
0234 A5 00 85 07   026C 3E E8 A2 04   02A4 27 58 2D 44
```

Dieses AIM-65/PC-100-Programm gestattet die Messung von Programmlaufzeiten bis zu 65 ms und beansprucht den Adressenraum 0200 bis 0298. An den Adressen 0299 bis 02A4 finden sich Konstanten

Noch einmal: Telefon-Modem

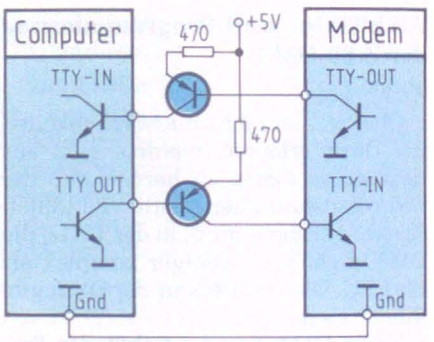
Das in Heft 10/1980 der FUNKSCHAU veröffentlichte Telefon-Modem besitzt sowohl eine TTL- als auch eine 20-mA-Schnittstelle (TTY-Stromschleife). Der Anschluß an Computer, die ebenfalls ein 20-mA-Schleifeninterface besitzen, ist wegen der bestehenden Pegelverhältnisse und der nicht galvanischen Trennung nur mit zwei zusätzlichen Transistoren möglich (*Bild*). Solche Computer sind z. B. AIM-65, PC-100, SYM-1 und KIM-1.

Die Übertragung von Basic-Programmen und Editortexten per Telefon ist bei AIM-65 und PC-100 recht einfach, wenn man das Lochstreifen-Format des Monitorprogramms verwendet. Bei der Aussendung ist wie folgt zu verfahren:

1. TTY-Schalter des AIM (S 3) auf „intern" stellen, Modem-Sender (Modulator) einschalten.
2. Datenausgabe per AIM-Monitorprogramm z. B. mit SAVE vorbereiten, so daß OUT= erscheint.
3. TTY-Schalter des AIM auf „TTY" stellen und auf der AIM-Tastatur die Taste L drücken. Sofort beginnt die Datenausgabe.
4. TTY-Schalter des AIM nach beendeter Übertragung auf „intern" zurückstellen, Reset drücken und mit 6 bzw. T in Basic bzw. Editor zurückgehen.

Vorher müssen allerdings die Zellen A417 und A418 mit den Daten 0C und C2 belegt werden, um die Übertragungsgeschwindigkeit auf 300 Bd einzustellen. Dies gilt auch für den Empfang von Daten mittels Modem:

1. Modem-Speicher aus- und TTY-Speicher auf „intern" schalten.
2. Einlesen z. B. mit LOAD vorbereiten, so daß IN= erscheint.



Anschluß des Telefon-Modems aus Heft 10 an Computer mit 20-mA-TTY-Schnittstelle

3. TTY-Schalter des AIM auf „TTY" stellen und auf der AIM-Tastatur die Taste L drücken; jetzt kann die Gegenstation mit der Aussendung beginnen.
4. Nach Beenden der Übertragung TTY-Schalter auf „intern" zurückstellen, Reset drücken und mit Taste 6/T in Basic/Editor zurückgeben.

Verwendet man den Mikrocomputer KIM-1 mit angeschlossenem Terminal, so empfiehlt es sich, den Modem-Eingang mit dem Terminal-Eingang (20-mA-Schleife) parallelzuschalten und den KIM-TTY-Eingang mit einem einfachen Umschalter wahlweise an den Modem-Ausgang oder den Terminal-Ausgang zu legen. Das Senden geschieht mit Taste Q (s. KIM-Handbuch), das Empfangen mit Taste L. Fe.

Hans-Georg Joepgen

Pascal auf einem Mikrocomputer

Erfahrungen mit dem UCSD-System des Apple

Die höhere Programmiersprache Pascal hat sich in der zweiten Hälfte des vergangenen Jahrzehnts als Bereicherung des Sprachenrepertoires großer Rechenanlagen durchgesetzt und auch Einzug in den Minicomputer-Bereich gehalten. Die durchweg günstigen Beurteilungen von Pascal ließen die Forderung immer lauter werden, auch dem Mikrocomputer-Anwender diese Sprache zu erschließen.

Die Hersteller der „Massen-Mikros" TRS-80, PET, Apple u.a. zeigten sich aus gutem Grund anfänglich durchaus nicht glücklich über Kundenwünsche, Pascal einzubauen. Als aber Drittunternehmen in den Vereinigten Staaten mit durchaus nicht vollkommenen Pascal-Erweiterungen zu nennenswerten Umsätzen kamen, erkannten auch die Massenrechner-Fabrikanten, daß ein weiteres Sträuben gegen den Mode-Trend nicht länger tunlich sei.

Wie man aus Amerika hört, befassen sich unterdessen sowohl Commodore als auch Tandy mit Pascal. Apple Computer Incorporated war der Schnellste unter den Marktführern: Seit dem Spätjahr 1979 bieten auch deutsche Apple-Fachhändler den vom Generalimporteur Basis Münster eingeführten Pascal-Erweiterungssatz zu Preisen um 1300 DM (in den USA: 495 Dollar) an. Was bekommt man fürs Geld? Einen solchen Pascal-Satz erprobte der Autor auf einem Apple II „Plus" und berichtet über die in zehnwöchigem Betrieb gewonnenen Erfahrungen.

Das Pascal-Paket: Hardware, Software, Literatur

Als die beiden wichtigsten Einführungswerke in Pascal gelten zur Zeit das „Pascal User Manual and Report" von Jensen-Wirth und Bowles' „Microcomputer Problem Solving Using Pascal" [1, 2]. Der Verfasser dieses Beitrags hatte sich zur Vorbereitung auf das angekündigte Pascal-Paket beide Werke besorgt und erlebte eine Überraschung: Beide Bücher lagen als Standard-Zubehör dem Pascal-Satz nochmal bei, in einer etwas verkleinerten, aber ansonsten ungekürzten Sonderausgabe der Originalwerke; dazu ein über 300seitiges Buch „Apple Pascal Reference Manual". Als Verfasser von Software und Dokumentation nennt dieses Handbuch rund drei Dutzend Namen, die Mehrzahl der Autoren Hochschullehrer und Studenten fortgeschrittener Semester am Institut für Informationswissenschaft der Hochschule von Kalifornien in San Diego; eine Arbeitsgruppe, die den von Apple gewählten Pascal-Dialekt („UCSD" – University of California at San Diego) unter Leitung von Kenneth L. Bowles entwickelte.

Weiter im Paket: Eine Einsteckkarte für den Steckplatz Nummer 0 des Rechners, sechs Disketten, zwei Lesespeicher-Bausteine, eine ausführliche, bebilderte Montageanleitung und Werkzeug zum IC-Wechseln. Der Umbau war in einer Viertelstunde geschehen. Nach dem Einlegen zweier Disketten in die Minilaufwerke und dem Einschalten von Rechner und Fernsehempfänger meldete sich Pascal mit seinem „Outer Command Level" auf dem Schirm, der übergeordneten Anweisungsebene, und fragte an, welchen Teil des Betriebssystems der Operator nun zu laden wünsche: Den Editor, Filer, Compiler, Linker, Assembler oder Debugger – Dienstprogramme, deren Namen selbst fortgeschrittene Mikrocomputer-Spezialisten bisweilen nur aus der Literatur kennen. So beeindruckend die vom Rechner angebotenen Möglichkeiten erscheinen, das Angebot ist nicht frei von Hochstapelei: Den Debugger enthielt unser Erprobungssystem noch nicht.

Laufwerkjustage höchst kritisch

Das Handbuch empfiehlt, so schnell wie möglich die wertvollen Systemdisketten zu duplizieren, die Originale an einem sicheren Ort zu verwahren und mit den Kopien weiterzuarbeiten. Neue Floppies müssen vor Gebrauch für die speziellen Pascal-Bedürfnisse gesondert initialisiert werden, die Aufzeichnungsdichte ist höher als im Standard-Disketten-Betriebssystem des Apple; deswegen auch der ROM-Tausch auf der Diskettenkarte. Obwohl das Handbuch ausführliche Anweisungen für diese Initialisierung gibt, gelang das „Formatieren" nicht. Nach mancherlei Versuchen stellte sich schließlich in tiefer Nacht heraus, daß Abweichungen von der Sollgeschwindigkeit der Antriebsmotore in den Disketten-Stationen die Ursache waren. Wäre nicht zufällig ein Justageprogramm für die Laufwerke eines gleichfalls im Hause befindlichen Rechners ITT 2020 zur Hand gewesen und hätte der Autor nicht den 2020 zum Neuabgleich der Apple-Laufwerke benutzt, dann wäre es nötig gewesen, Kundendienst-Techniker zu Hilfe zu rufen. Allerdings war der Kummer damit noch nicht ausgestanden. Dann begann der Pascal-Apple nämlich, schamlos zu lügen:

Zur Berichterstattung aufgefordert, welche Peripheriegeräte er als angeschlossen betrachte (das V-Kommando des Filers reportiert die „Volumes-On-

Line"), setzte der Rechner zwar nacheinander beide Laufwerke in Gang, behauptete aber, er hätte nur Verbindung mit einem einzigen Laufwerk. Des Rätsels Lösung hier: Zentrierfehler der Scheibe. Durch Neueinlegen, Klappenöffnen bei laufendem Antriebsmotor und Zupfen an der Diskette ließ sich das Laufwerk aber schließlich doch noch dazu bewegen, mit dem Rechner zusammenzuarbeiten – ein Fehler, der mit fortschreitender Betriebsdauer dann immer seltener auftrat.

Damit hat die Hardware-Schelte jedoch auch schon ihr Ende gefunden, weitere Beanstandungen diesbezüglich ergaben sich nicht. Was sonst noch am Pascal-Apple zu bemängeln ist, geht auf das Konto Software.

Die Subsysteme der Betriebssoftware

Die von der Meldezeile der übergeordneten Anweisungsebene angebotenen Subsysteme der Betriebssoftware können aus Platzgründen hier nur gestreift werden. Der Editor dient dem Arbeiten an den „Textfiles", dem Programm-Schreiben und -Ändern also. Er wird mit der Taste „E" aufgerufen und meldet sich dann per eigener Meldezeile, die Handreichungen anbietet wie Einfügen, Entfernen, Aufsuchen, Verändern und Vertauschen von Textteilen. Einige dieser Dienstleistungen haben ihrerseits eigene Meldezeilen, wobei für jede Ebene ein eigenes Vokabular existiert.

Dies soll an der F-Taste verdeutlicht werden: Das Betätigen von „F" im übergeordneten Anweisungsbereich bewirkt, daß sich der Rechner den Filer von der Diskette holt, im Editor jedoch, daß die Dienstleistung „Aufsuchen einer Textstelle mit dem Cursor" („Find") zur Verfügung gestellt wird. Neuerliches Betätigen von „F" danach kann beispielsweise der Bezeichnung der aufzusuchenden Textstelle dienen.

In den Subsystemen Filer, Compiler und so fort kommt den gleichen Tasten wieder eine völlig andere Bedeutung zu – höchst kompliziert und in eigenartigem Kontext zu der oft gehörten Behauptung, die besondere Stärke von Pascal liege in ihrer leichten Erlernbarkeit durch Anfänger. Freilich muß man aus Gründen der Fairness hier zweierlei anfügen: Apple-Pascal hilft durch kräftiges Soufflieren per Schirmmitteilung durch den Dschungel des Anweisungsvokabulars. Und niemand ist gezwungen, sich die Beherrschung auch noch der letzten und trickreichsten Dienstleistungen, die der Rechner anbietet, zu eigen zu machen.

Der Verfasser benutzte beim Programmieren von den Editor-Handreichungen nur die elementaren; das brachte zwar gelegentlich ein wenig mehr Tipparbeit, hielt aber den Kopf frei für Wesentliches. Nach Abschluß der Programmeingabe oder Textänderung bietet der Rechner verschiedene Wege zum Verlassen des Editors an: Verzicht auf Aufzeichnung des neuen Textes, Aufzeichnung in einem „Workfile" genannten Bereich der Systemdisketten oder Deponieren unter einem frei gewählten Namen auf einer beliebigen Scheibe.

Der Compiler übersetzt ein Programm-Textfile in einen Zwischencode, der im Speicher oder auf Diskette abgelegt wird und den bei Programmausführung ein Interpreter in die 6502-Maschinensprache des Apple umsetzt. Der Filer erlaubt die Kontrolle von Peripheriegeräten, das En-Bloc-Bearbeiten von Programm-Text und Programm-Code-Aufzeichnungen, Formatieren und Kopieren von Disketten, Setzen des Datums und noch vieles andere; der Linker dient dem Kombinieren von Code-Aufzeichnungen und dem Einbinden von Assembler-Segmenten oder Bibliotheksprogrammen in ein Benutzerprogramm. Alles in allem: Zwar sicherlich kleine Fische für Leute, die seit Jahren mit Großcomputern auf Du und Du stehen: Aus der Sicht des Mikrocomputer-Mannes hat es jedoch den Anschein, als ob allein bis zu einer befriedigenden Beherrschung des Betriebssystems von Pascal-Maschinen mehr an Lernbereitschaft aufgezehrt wird als beim Einarbeiten in eine komplette Mikrocomputer-Hochsprache wie Mikrosoft-Basic. Wer freilich bereit ist, zu büffeln und zu üben, der wird nach möglicherweise bitteren und frustrierenden Anfängernöten den Komfort von UCSD-Pascal auf dem Apple zu schätzen lernen. Doch weiter im Erfahrungsbericht: Was ist über die Programmiersprache selbst und über die Betriebserfahrungen beim Abarbeiten von Programmen zu bemerken?

Nach Anfangsenttäuschungen: Die „große Liebe" keimt

Die Freunde dieser Sprache rühmen an Pascal die bequemen Möglichkeiten, je nach Wunsch und Aufgabenstellung eigene Datentypen mit trickreich den jeweiligen Problemen angepaßten Strukturen einzuführen, sagen darüber hinaus, daß es allein schon Syntax und Grammatik der Sprache an sich, quasi selbsttätig, bewirken, daß ein Pascalprogramm wohlstrukturiert, wartungsfreundlich, selbstdokumentierend und elegant gerät. All das ist *im Grundsatz* durch die Betriebserfahrungen mit Apple-Pascal durchaus bestätigt worden. Beim Verfasser begann nach einer durch überhöhte Anfangserwartungen ausgelösten „Frustperiode" allmählich die große Pascal-Liebe zu keimen; er mag die Sprache nicht mehr missen. Im Grundsatz, hieß es oben: Und in der Tat – die Vorteile selbstgeschaffener Datentypen kommen bei den heute auf Mikrocomputern gefahrenen Programmen zumeist nicht voll zum Tragen, hier setzt schon der zumeist enge Speicherraum natürliche Grenzen. Und was den pascal-immanenten Zwang zu sauberem Programmieren angeht: Es ist in Pascal durchaus möglich, ein Programm von Flickteppich-Struktur zu schreiben und zum Laufen zu bringen – nur, die Versuchung, so etwas zu tun, ist geringer als in Basic.

Pascal – das „große Diskettentheater"

Niklaus Wirths Standard-Pascal und erst recht die kräftig erweiterte UCSD-Version dieser Sprache sind nicht von Haus aus dazu geschaffen, sich auf Mikrocomputern wohlzufühlen, sondern für großzügiger bemessene Verhältnisse angelegt. Wiewohl das Pascal-Paket einen 48-kByte-Apple zur 64-K-Maschine macht, findet immer nur ein Teil der Betriebssoftware im Speicher Platz. Das bedeutet, daß die Disketten nahezu pausenlos in Betrieb sind, wenn man bei der Arbeit die Kommando-Ebenen wechselt. Das dauert jeweils zwar nur Sekunden, summiert sich aber, während der Operator wartet; zwischendurch sind, selbst beim Betrieb mit zwei Laufwerken, zusätzlich noch Disketten von Hand zu wechseln, etwa dann, wenn man auf die (übrigens reichhaltige) Unterprogrammbibliothek zurückgreifen will, um Terminalbetrieb über

eine Konsole abzuwickeln oder Grafiken (in Farbe und mit hoher Auflösung) einzusetzen.

Am folgenden Beispiel soll gezeigt werden, gegen welche Überraschungen man sich zu wappnen hat, wer von einer Interpreter-Sprache zur Compiler-Sprache Pascal umsteigt – wir betrachten den Fall, daß ein Programmierer versehentlich einen Doppelpunkt tippte, als er ein Semikolon meinte. Apple-Basic wird sich gleich bei der Zeicheneingabe beschweren, Microsoft-Dialekte wie Applesoft, Commodore Basic oder Siemens PC 100 Basic melden sich bei erstmaligem Programmdurchlauf mit ausführlicher Fehlermitteilung und teilen die Zeilennummer mit. Man listet die Zeile, repariert den kleinen Schaden, und die Welt ist wieder in Ordnung. Was steht in einem solchen Fall dagegen einem Pascal-Jünger ins Haus? Das Programm ist, einschließlich des vertrackten Fehlers, fertig; man verabschiedet den Editor, die Textfile wird auf Diskette geschrieben. Jetzt brauchen wir den Compiler, der wird nachgeladen. Das System compiliert vor sich hin und entdeckt nun den Fehler. Von der Diskette wird die Fehlerbehandlungsroutine geholt, auf dem Schirm erscheint die Fehlerbeschreibung, der Rechner bietet den Editor an, läßt aber auch die Möglichkeit offen, weiter zu compilieren, um zusätzliche Fehler zu suchen. Ob es noch welche gibt, wissen in diesem Augenblick weder der Rechner noch sein Meister. Also holt man den Editor von Diskette, den Cursor positioniert der Apple freundlicherweise selber auf oder zumindest in der Nähe des Fehlers. Nun verbessert man, verabschiedet den Editor, der Text kommt erneut auf Diskette, man holt den Compiler – und wenn kein weiterer Fehler im Programm war, bekommt man jetzt ein vollziehbares Programm in Zwischencode: Oh, wie war dereinst vordem – das Basic-Leben so bequem!

So lästig und ärgerlich derlei Umständlichkeiten sind: Von einer gewissen Programmgröße an, Größenordnung 8 Kilobyte, nimmt man sie beim Programmieren gern in Kauf. Die saubere Trennung zwischen lokalen und globalen Variablen, die strenge Ordnung beim Einsatz der in Pascal nach „Procedures" und „Functions" sortierten Unterprogramme helfen, Überblick zu behalten, bewahren vor der Gefahr, sich in einem Wald zu verirren, den man vor lauter Bäumen nicht mehr erkennt. Wenn der Verfasser an gewisse umfangreiche Basic-Projekte und die damit verbundenen Schwierigkeiten [3, 4] zurückdenkt, so kann der Stoßseufzer nur mit Mühe unterdrückt werden: Ach, hätte man damals doch nur schon Pascal gehabt!

Schlußurteil

Kommen wir zum Schlußurteil. Der Hardware-Aufwand, der bereits vor dem Umrüsten vorhanden sein muß, ist erheblich: 48 KByte RAM, zwei Disketten-Laufwerke. Zwar ist auch ein Ein-Diskettenbetrieb möglich, doch kann ein solches Verfahren nur Zeitgenossen mit viel Zeit und einer Engelsgeduld empfohlen werden. Saftig ist weiter der Peis für das Pascal-Paket; freilich, der erscheint angemessen, wenn man in Rechnung stellt, wie viele qualifizierte Träger bekannter Namen laut Autorenverzeichnis des Handbuches damit befaßt waren, dem Apple-Pascal Reife zu verleihen. Angemessen erscheint der Preis darüber hinaus zusätzlich, wenn man in die Bewertung einbezieht, wie liebevoll, sorgfältig, redlich und umfangreich die Dokumentation geriet. Und einzubeziehen bei der Betrachtung des Kosten/Nutzen-Verhältnisses ist weiter, daß die Pascal-Erweiterungskarte streng genommen gar keine Pascal-Erweiterungskarte ist, sondern eigentlich eine universelle Sprachenkarte, die erst durch die Systemdisketten zum Leben erweckt wird, sich aber genausogut durch andere Disketten zum Umgang mit weiteren Hochsprachen bewegen läßt: Apple hat angekündigt, daß weitere Sprachen folgen werden.

Was bekommt man fürs Geld? Ein volltaugliches Pascal-System (freilich geringer Rechengenauigkeit: Beim Präzisionstest „Kommissar" [5] stieg der Apple bereits nach der 361. Iteration aus, der PET dagegen bringt's auf 886). Pascal-Programme laufen schneller als Basic-Programme, freilich auch deutlich langsamer als Assembler-Programme [6]. Fazit: Für Leute, die mit einem wichtigen Trend in der Mikro-Szene vertraut werden wollen, ist Apple-Pascal als Zweitsprache neben einer interpretierten Hochsprache durchaus empfehlenswert. Freilich möge man sich vor überzogenen Erwartungen hüten: Pascal ist schließlich eine Computersprache aus den Anfängen des vergangenen Jahrzehnts, wesentlich betagter somit als beispielsweise Microsoft-Dialekte, und das merkt man. Pascal – die Sprache der Zukunft? Wohl kaum. Sicherlich aber eine passable Grundlage für eine solche Programmiersprache der Zukunft.

Literatur

[1] *Jensen, Kathleen,* und *Wirth, Niklaus:* „Pascal User Manual and Report", Second Corrected Reprint of the Second Edition, Springer-Verlag, New York–Heidelberg–Berlin, 1978.

[2] *Bowles, Kenneth L.:* „Microcomputer Problem Solving Using Pascal", Springer-Verlag, New York–Heidelberg–Berlin, 1977.

[3] *Joepgen, Hans-Georg:* „Komfort-Programm zum Rechnen mit komplexen Zahlen", ELEKTRONIK 1979, Heft 17. Franzis, München.

[4] *Joepgen, Hans-Georg:* „Basic-Programm simuliert Wobbel-Meßplatz", ELEKTRONIK 1980, Heft 2, Franzis, München.

[5] *Joepgen, Hans-Georg:* „‚Kommissar' deckt Rechenungenauigkeit auf", Hobbycomputer 1, Sonderheft von FUNKSCHAU, ELO und ELEKTRONIK, Franzis, München, 1978 und 1979.

[6] *Joepgen, Hans-Georg:* „Suchprogramm – formuliert in drei Sprachen", Hobbycomputer 2, Sonderheft der FUNKSCHAU, Franzis, München, 1979.

[7] *Hofer, Rudolf:* „Brauchen wir Pascal?", FUNKSCHAU 1980, Heft 2, Franzis, München.

[8] *Wirth, Niklaus:* „Algorithms plus Data Structures = Programs", Prentice Hall, Englewood Cliffs, New Jersey, USA, 1976.

[9] *Forsyth, Charles H.,* und *Randall, Howard:* „Compilation and Pascal on the New Microprocessors". Byte 1978, Heft 8, Byte Publications, Peterborrough, New Hampshire, USA.

[10] *Schauer, Helmut:* „Pascal für Anfänger", Dritte Auflage, R. Oldenbourg Verlag, Wien und München, 1979.

[11] *Wilson, I. R.,* und *Addyman, A. M.:* „Leichtverständliche Einführung in das Programmieren mit Pascal", Carl Hanser Verlag, München und Wien, 1979.

[12] *Lübbert, William F.:* „What's Where in the Apple", Micro – the Magazine for the Apple, KIM, PET. 1979, Nr. 15. Europa-Vertrieb: Micro Shop Bodensee, D 7778 Markdorf.

Michael Konz

AIM 65 plottet Funktionen

Wer häufig Funktionen zeichnen muß, der wird sich gelegentlich einen Plotter wünschen. Leider sind vernünftige x-y-Plotter auch heute noch recht teuer. Die Verwendung eines Bildschirmterminals ist aufgrund seiner geringen Auflösung unbefriedigend. Der Thermodrucker des AIM-65 gibt aber zusammen mit etwas Software einen für viele Anwendungen recht brauchbaren Plotter ab, und das ohne Änderung der Hardware.

Das Programm, bestehend aus einem in Basic geschriebenen Teil, der die Berechnung der Funktionswerte übernimmt, und einem Segment im Maschinencode (*Bild 1*), das den Druckvorgang steuert, liefert ein Raster von 100 x 100 Punkten.

Die darzustellende Funktion wird in Zeile 100 in das Basic-Programm (*Bild 2*) eingefügt. Dabei ist zu beachten, daß die Zeile für die x-Werte übersprungen wird, bei denen sonst ein Fehler auftreten würde. Das sieht dann z.B. so aus:
99 IF X=0 THEN 110
100 Y=1/X

Interessant ist nun die Möglichkeit, durch Änderung der Intervalle, in deren Bereich die Funktion berechnet und gedruckt wird, Ausschnittsvergrößerungen der Kurve zu erstellen. Einen Beispielausdruck zeigt *Bild 3*.

Eine Darstellung von Relationen ist mit diesem Programm leider nicht möglich. In einer Tabelle an den Adressen 0F00...0F63 werden die zu jeder Abzisse gehörigen Ordinaten gespeichert. Daher kann jedem x-Wert nur ein y-Wert zugeordnet werden, also maximal 100 Punkte.

Um beide Programmsegmente gleichzeitig auf Kassette speichern zu können, empfiehlt es sich, die Monitor-Dump-Routine zu benutzen. Beginn und Ende des Basic-Teiles erfährt man von den Speicherzellen 73/74 bzw. 75/76. Nachdem die beiden Programmsegmente übertragen wurden, muß noch der Bereich von 73...7A auf Band kopiert werden.

Um das Programm anschließend wieder einzulesen, lädt man es einfach nach Initialisieren des Basic-Interpreters mit der entsprechenden Routine des Monitors.

```
0C00 A9 FF 8D 74 | 0C84 C0 14 F0 10
0C04 A4 A9 63 8D | 0C88 A9 80 8D 08
0C08 76 A4 A9 63 | 0C8C A8 A9 09 8D
0C0C 8D 77 A4 20 | 0C90 09 A8 20 1B
0C10 BD 0C A9 C1 | 0C94 EC 4C 30 0C
0C14 8D 0C A8 20 | 0C98 4C 24 0C EA
0C18 A0 FF D0 08 | 0C9C 20 AF 0C A2
0C1C 20 A0 FF D0 | 0CA0 0A 20 A0 FF
0C20 03 4C 79 F0 | 0CA4 CA 10 FA A9
0C24 A0 00 8C 01 | 0CA8 E1 8D 0C A8
0C28 A8 A9 02 2C | 0CAC 4C CB F0 A9
0C2C 0D A8 F0 FB | 0CB0 00 8D 01 A8
0C30 AD 78 A4 8D | 0CB4 AD 00 A8 29
0C34 01 A8 AD 00 | 0CB8 FC 8D 00 A8
0C38 A8 0D 79 A4 | 0CBC 60 A9 00 8D
0C3C 8D 00 A8 A9 | 0CC0 7B A4 8D 78
0C40 A4 8D 08 A8 | 0CC4 A4 8D 79 A4
0C44 A9 08 8D 09 | 0CC8 A9 01 8D 7A
0C48 A8 AE 76 A4 | 0CCC A4 AE 76 A4
0C4C 2C 74 A4 10 | 0CD0 BD 00 0F CD
0C50 0B CA E0 59 | 0CD4 77 A4 D0 20
0C54 D0 0F EE 74 | 0CD8 AD 78 A4 0D
0C58 A4 E8 D0 09 | 0CDC 7A A4 8D 78
0C5C E8 E0 64 D0 | 0CE0 A4 AD 79 A4
0C60 04 CE 74 A4 | 0CE4 0D 7B A4 8D
0C64 CA 8E 76 A4 | 0CE8 79 A4 0E 7A
0C68 C8 C0 0A F0 | 0CEC A4 2E 7B A4
0C6C 04 C0 14 D0 | 0CF0 8A 38 E9 0A
0C70 0A CE 77 A4 | 0CF4 AA B0 D9 60
0C74 AD 77 A4 C9 | 0CF8 AD 77 A4 C9
0C78 FF F0 20 20 | 0CFC 32 F0 D9 E0
0C7C BD 0C 20 1B | 0D00 32 F0 D5 D0
0C80 EC 20 AF 0C | 0D04 E5
```

Bild 1. Maschinenprogrammteil für das Plotten mit dem Thermodrucker des AIM-65

```
1 REM ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
2 REM ◆◆◆◆◆PLOT◆◆◆◆◆
3 REM ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
4 REM MICHAEL KONZ, 7.11.1979
10 PRINT!" "
11 INPUT!"X-INT=";A,B
20 IF A>B THEN 11
30 E=(B-A)/100
35 N=-A/E:P=N
40 IF N<0 OR N>255 THEN P=255
50 POKE 3328,P
60 INPUT!"Y-INT=";C,D
70 IF C>D THEN 60
80 F=(D-C)/100
82 M=-C/F:P=M
84 IF M<0 OR M>255 THEN P=255
86 POKE 3324,P
90 FOR X=A TO B STEP E
100 Y=EXP(X)◆COS(X)
110 IF Y<C OR Y>D THEN Y=(255-M)◆F
120 POKE INT(X/E+N+.5)+3840,INT(Y/F+M+.5)
130 NEXT
140 POKE 4,0
145 POKE 5,12
150 X=USR(0)
170 PRINT!" "
180 END
```

Bild 2. Das Basic-Programm enthält die zu plottende Funktion in Zeile 100

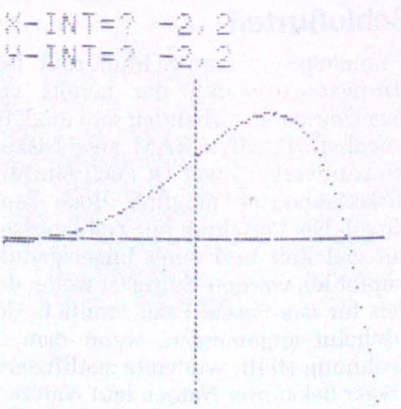


Bild 3. Beispielausdruck mit dem AIM-65

Peter Hanitzsch

TI-59 berechnet Extremwerte von Funktionen

Ein bei der Optimierung von Schaltungen und anderen Entwicklungsaufgaben oft auftretendes Problem ist die Berechnung von Funktions-Extremwerten, z. B. um die günstigsten Bauelementewerte herauszufinden, nachdem man den Zusammenhang zwischen ihrer Dimensionierung und den Schaltungseigenschaften als mathematische Funktion bestimmte. Ein programmierbarer Taschenrechner löst dieses Problem mit einem Minimum an Zeitaufwand.

Das hier beschriebene Programm sucht eventuell vorhandene Extremwerte (Minima und Maxima) einer Funktion $f:]a, b[\rightarrow IR$ im *offenen* Intervall $]a, b[$. Dabei wird zuerst geprüft, ob überhaupt Extrema vorhanden sind. Ist dies der Fall, wird die Art des Extremums angezeigt; in der Anzeige erscheint eine 1, falls ein Minimum und eine 2, falls ein Maximum vorliegt. Erscheint eine 0, konnte in $]a, b[$ keine Extremstelle der vom Anwender definierten Funktion ermittelt werden. Besteht nur Interesse für die Art des Extremums, kann nach jeder Anzeige der Suchlauf durch Drücken von R/S fortgesetzt werden, bis schließlich 0 angezeigt wird. Möchte man die Abszisse x_e des Extremums angezeigt bekommen, drücke man Taste E. Hierbei wird gleichzeitig die Schrittweite Δx halbiert. Drückt man nun Taste D, so erhält man eine neue Näherung von x_e, die über Taste E angezeigt wird. Eine Wiederholung dieses Verfahrens (Taste E, Taste D) verbessert stets die Genauigkeit von x_e, wobei der Fehler kleiner als Δx ist. Möchte man den zugehörigen Funktionswert y_e errechnen, so drücke man nach E die Taste C'. Soll der Suchlauf mit der ursprünglich eingegebenen Schrittweite fortgesetzt werden, drücke man D' *(Tabelle 1)*. Die Grenzen des Extremwertintervalles befinden sich in den Speichern 59 und 53.

Der Programmablauf wird in *Bild 1* deutlich. Die Eingabe von f erfolgt ab Programmschritt 140. Bei der Anfangsverteilung stehen die Speicher 00 bis 50 und alle Klammerebenen zur Verfügung. Abgeschlossen wird die Eingabe von f durch INV SBR. Flag 8 wurde gesetzt, um das Programm bei einem möglichen Fehler zu stoppen. Ein Beispiel: $f(x) = x^3$, $]a, b[=]-1, +1[$, Eingabe von f mittels y^x ! Hier sollte f wie folgt eingegeben werden:

GTO C' LRN (,STO 00,x^2,x,RCL 00,) INV SBR LRN

Man beachte stets, daß f in dem *abgeschlossenen* Intervall $]a, b[$ definiert sein muß! *Bild 2* zeigt das Programmlisting, und *Tabelle 2* nennt ein Rechenbeispiel.

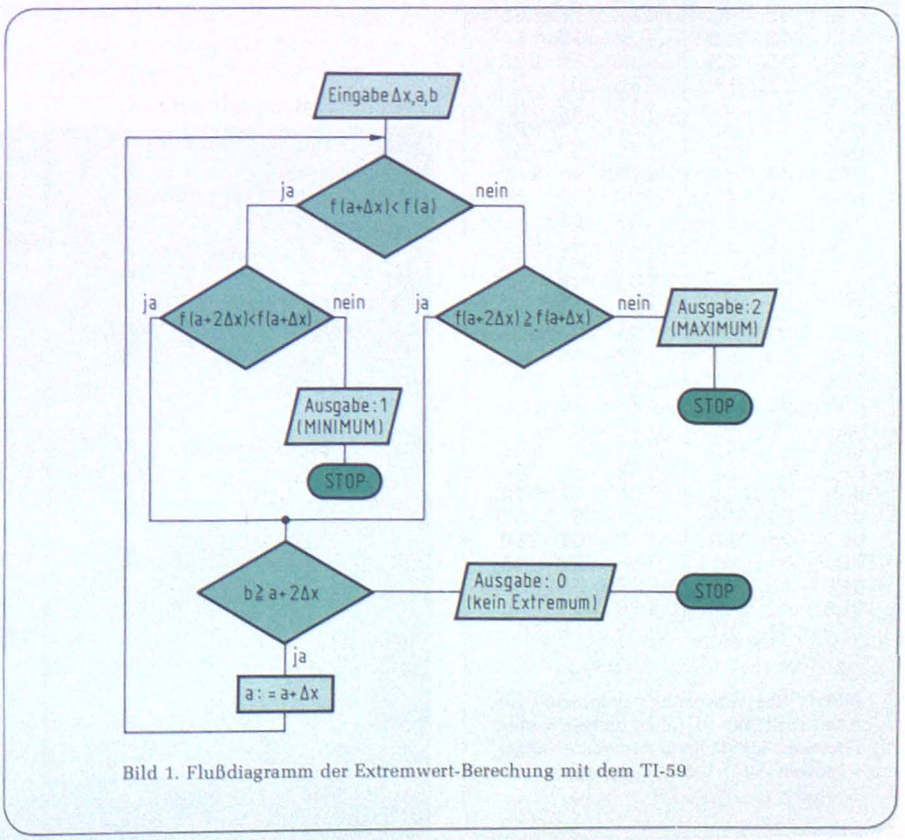


Bild 1. Flußdiagramm der Extremwert-Berechung mit dem TI-59

000	53	(	070	53	(
001	43	RCL	071	43	RCL
002	52	52	072	54	54
003	75	-	073	75	-
004	43	RCL	074	43	RCL
005	54	54	075	51	51
006	54	)	076	54	)
007	92	RTN	077	22	INV
008	71	SBR	078	77	GE
009	00	00	079	00	00
010	00	00	080	08	08
011	22	INV	081	61	GTO
012	77	GE	082	00	00
013	00	00	083	20	20
014	84	84	084	43	RCL
015	01	1	085	58	58
016	91	R/S	086	75	-
017	61	GTO	087	43	RCL
018	00	00	088	53	53
019	28	28	089	95	=
020	71	SBR	090	77	GE
021	00	00	091	00	00
022	00	00	092	28	28
023	77	GE	093	00	0
024	00	00	094	91	R/S
025	84	84	095	76	LBL
026	02	2	096	11	A
027	91	R/S	097	42	STO
028	43	RCL	098	59	59
029	57	57	099	91	R/S
030	44	SUM	100	76	LBL
031	59	59	101	12	B
032	76	LBL	102	42	STO
033	14	D	103	58	58
034	86	STF	104	91	R/S
035	08	08	105	76	LBL
036	29	CP	106	13	C
037	43	RCL	107	42	STO
038	59	59	108	57	57
039	85	+	109	42	STO
040	43	RCL	110	56	56
041	57	57	111	91	R/S
042	95	=	112	76	LBL
043	42	STO	113	15	E
044	55	55	114	25	CLR
045	71	SBR	115	02	2
046	01	01	116	22	INV
047	40	40	117	49	PRD
048	42	STO	118	57	57
049	54	54	119	43	RCL
050	43	RCL	120	59	59
051	55	55	121	85	+
052	85	+	122	43	RCL
053	43	RCL	123	53	53
054	57	57	124	95	=
055	95	=	125	55	÷
056	42	STO	126	02	2
057	53	53	127	95	=
058	71	SBR	128	91	R/S
059	01	01	129	76	LBL
060	40	40	130	19	D'
061	42	STO	131	43	RCL
062	52	52	132	56	56
063	43	RCL	133	42	STO
064	59	59	134	57	57
065	71	SBR	135	61	GTO
066	01	01	136	00	00
067	40	40	137	28	28
068	42	STO	138	76	LBL
069	51	51	139	18	C'

Bild 2. Das vollständige Programm. Ab dem Schritt 140 ist die zu untersuchende Funktion als Unterprogramm – abgeschlossen mit INV SBR – einzugeben

Tabelle 1: Benutzerinstruktionen

Schritt	*Eingabe*	*Taste*	*Anzeige*
1) Eingabe des Programms bzw. Einlesen von Seite 1 der Magnetkarte	0	–	1.
2) Eingabe der Funktion f		GTO C'	
		LRN	14000
		.	.
		.	.
		.	.
		.	.
		.	.
		.	.
		.	.
		INV SBR	...92
		LRN	
3) Eingabe der Intervallgrenzen und der Schrittweite	a	A	a
	b	B	b
	Δx	C	Δx
4) Berechnung der Extrema			
a_1) Programmstart	–	D	1,2 oder 0
a_2) Fortsetzung des Suchlaufs	–	R/S	1,2 oder 0
b_1) Anzeige von x_e und Halbierung von Δx	–	E	x_e
b_2) Neue Näherung von x_e	–	D	1 oder 2
b_3) Anzeige des neuen Näherungswertes und erneute Halbierung von Δx		E	x_e
Für weitere Näherungen von x_e gehe zu 4 b_2)			
c) Fortsetzung des Suchlaufs mit c) ursprünglichem Δx		D'	1,2 oder 0
d) Berechnung von $y_e = f(x_e)$	x_e	C'	y_e

Tabelle 2: Beispielrechnung

Bestimmung der Extrema der Funktion
$f(x) := \sin x - \sqrt{3} \cos x$
im Intervall $]0°, 720°[$. (Schrittweite $\Delta x = 30°$)
Eingabe der Funktion f:
GTO C' LRN (, STO 00, SIN, –, 3, $\sqrt{x}$, x, RCL 00, COS,) INV SBR LRN

Eingabe	*Taste*	*Anzeige*	
0	A	0	
720	B	720	
30	C	30	
	D	2	(32,8 Sekunden)
	E	150	*Maximum* bei $x_{e1} = 150°$
	C'	2	$f(150°) = 2 = y_{e1}$
Fortsetzung des Suchlaufs mit $\Delta x = 30°$	D'	1	(40 Sekunden)
	E	330	*Minimum* bei $x_{e2} = 330°$
	C'	–2	$f(330°) = -2 = y_{e2}$
Fortsetzung des Suchlaufs mit $\Delta x = 30°$	D'	2	(40 Sekunden)
	E	510	*Maximum* bei $x_{e3} = 510°$
	C'	2	$f(510°) = 2 = y_{e3}$
Fortsetzung des Suchlaufs mit $\Delta x = 30°$	D'	1	(42 Sekunden)
	E	690	*Minimum* bei $x_{e4} = 690°$
	C'	–2	$f(690°) = 2 = y_{e4}$
Fortsetzung des Suchlaufs mit $\Delta x = 30°$	D'	0	(7 Sekunden)
			Keine weiteren Extrema

Somit erhält man in 161,8 Sekunden folgende sämtlichen Extrema von f:
Ma (150°, 2), Mi (330°, –2), Ma (510°, 2), Mi (690°, –2) im Intervall $]0°, 720°[$.

SGS-Nanocomputer: Monitorprogramm-Listing jetzt lieferbar

Auf das Drängen der FUNKSCHAU-Redaktion hin hat sich SGS-Ates nun entschlossen, ein Listing des Nanocomputer-Monitorprogramms (siehe Heft 5/1980, Seite 83) herauszugeben. Dem war ein Schriftwechsel vorausgegangen, den wir im folgenden gekürzt wiedergeben.

An: FUNKSCHAU-Redaktion *11.3.1980*

In dem genannten Artikel schreibt R. Hofer, der hohe Preis sei durch das Lehrmaterial gerechtfertigt: zwei Bücher in deutscher Sprache, von denen das zweite sich vorwiegend mit Interfacing befasse. Es kostete uns fünf Minuten zu recherchieren, daß dieser zweite Band noch nicht existiert. Hier wird ein System hochgelobt, dessen wichtigste Teile noch nicht existieren.

E. Kallabis, Dr. F. Kiekheben

An: Fa. SGS-Ates *14.3.1980*

Anbei erhalten Sie eine Kopie eines Briefes zweier Leser, die sich bitter darüber beklagen, daß in unserem Nanocomputer-Bericht die „Unwahrheit" verbreitet wird. Ich möchte auch erwähnen, daß mehrere Leser bemängeln, daß SGS nicht einmal ein Monitorprogramm-Listing herausgibt. Ich kann niemandem den Kauf eines Computers empfehlen, der mit einer so schlechten Software-Unterstützung geliefert wird.

H. Feichtinger,
FUNKSCHAU-Redaktion

An: FUNKSCHAU-Redaktion *22.4.1980 (!)*

Für die Nanocomputer-Lehrbuchreihe sind insgesamt drei Ausgaben geplant. Band 1: Z-80-Einführung und Programmierung; Band 2: Digitaltechnik; Band 3: Z-80-Interfacetechnik und Anwendung. Ab Anfang Mai gibt es Band 3 in deutscher Sprache, Band 2 voraussichtlich ab Januar 1981. Jedem Nanocomputer liegt eine ausführliche Beschreibung der dem Anwender zugänglichen Unterprogramme der Betriebssoftware bei, mit genauer Angabe der Einsprungadresse und der verwendeten CPU-Register. Ab Mitte 1980 ist außerdem auf Wunsch das komplette Nanocomputer-Programmlisting verfügbar.

L. May,
SGS-Ates Deutschland

Meine Meinung...

Mehr Sorgfalt bitte!

Wenn man heute gewisse Fachveröffentlichungen aus den fünfziger Jahren zur Hand nimmt, mit denen die damals noch allgemein in „Röhrenkategorien" denkenden Rundfunkmechaniker und Radiobastler auf das heraufziehende Zeitalter des Transistors vorbereitet werden sollten, dann muß man sich wundern – darüber, daß es trotz derartiger Werke doch Zehntausende von Profis und Amateuren innerhalb so kurzer Zeit schafften, die neuen Bauteile mit ihrer ungewohnten Schaltungstechnik in den Griff zu bekommen: So didaktisch ungekonnt, voller Ballast und Umständlichkeiten und bisweilen sogar grob unrichtig war vieles, das damals Fachleute für Fachleute schrieben.

Betrachtet man nun einen Teil dessen, was heute an Fachbüchern zur Mikrocomputertechnik erscheint, dann regt sich der Verdacht, eine künftige Lesergeneration könne in 20 oder 30 Jahren mit ähnlichem Mitleid auf uns zurückblicken, die wir uns in diesen Monaten mit den „intelligenten Winzlingen" auseinanderzusetzen haben. So hilfreich die heute in bunter Vielfalt angebotenen Fachveröffentlichungen auch sein mögen – viele von ihnen enthalten manches, das dem Anfänger den Zugang zur Mikrocomputerwelt eher erschwert denn erleichtert. Es soll an dieser Stelle nicht die Rede sein von den eilig auf den Markt gebrachten Erzeugnissen halbseriöser Verlage, zu denen beispielsweise ein Programmierhandbuch zählt, mit dessen Hilfe man den Eingabe-Ausgabe-Baustein seines Rechners zuverlässig zerstören kann, wenn man den dort gegebenen Hinweisen folgt; nicht die Rede sein soll weiter von den hanebüchenen Übersetzungen amerikanischer Werke mit ihrem hohen Unterhaltungswert, den sie mißglückten Eindeutschungsversuchen verdanken.

Betrachten wir allein das im großen ganzen zufriedenstellende Dokumentationsmaterial angesehener Firmen, so bleibt genug Grund zur Klage. Firmenunterlagen zum Befehlssatz der CPU 6502 sprechen zum Beispiel von einem „Transfer" (TAX, TAY, TXA und so fort) eines Bytes von Register zu Register. Nun gehört es in allen Kultursprachen zum Wesen eines „Transfers", daß sich die transferierte Sache nicht mehr am alten Ort, sondern an einem neuen Platz befindet. Genau das aber trifft bei den T-Befehlen durchaus nicht zu, das Byte am Ausgangsort bleibt unverändert: Es wird lediglich zum Zielort kopiert. Sinngemäß das gleiche gilt für die „Move-Befehle" der CPU 8080 und ihrer Nachfolger. Ein weiterer Schnitzer: Da heißt es in einer weitverbreiteten Firmenschrift, die Adressierungsart gebe an, auf welche Weise die CPU auf den externen Speicher zugreife. Wenige Zeilen weiter ist die Rede von der „Adressierungsart ‚impliziert' (implied)" – bei der bekanntlich überhaupt nicht auf Speicher außerhalb der CPU zurückgegriffen wird. Ein letztes Beispiel schließlich: Im vorläufigen Handbuch eines von einem deutschen Konzern verkauften Mikrocomputers wird unbefangen von „Thermoelementen" geplaudert. Gemeint sind freilich die Druckelemente eines Thermodruckers.

Alle diese Exempel zeigen eines: Druckschriften nehmen oft zu wenig Rücksicht auf den Leser, ihre Verfasser oder Übersetzer versäumen es, sich in die Rolle eines Lesers hineinzudenken, der sich in ein neues und schwieriges Fachgebiet einarbeiten will. Denn wenn sich Widersprüche auftun, dann wird ein solcher Leser wohl zuerst einmal an sich zweifeln, häufig zurückblättern und zu der Meinung kommen, er selbst habe etwas falsch verstanden: Ehe er dem Fachmann zu mißtrauen wagt. Gleichgültig, ob Schlamperei oder didaktischer Dilettantismus die Ursachen für derlei Schnitzer sind: Sie erschweren völlig unnötigerweise den Zugang zu einem Fachgebiet zusätzlich, das ohnehin zu den abstraktesten Technologien zählt, die Menschen bisher hervorgebracht haben. Und wenn das so ist, so muß man von den Verfassern und Bearbeitern derartiger Werke mit gutem Recht verlangen, sich ihrer Verantwortung besser bewußt zu werden: Mehr Sorgfalt bitte!

Hans-Georg Joepgen

Primfaktorzerlegung für Eilige

In Heft 5/1980 wurde auf Seite 82 ein TI-59-Programm zur Zerlegung einer Zahl in ihre Primfaktoren abgedruckt. Das dabei gewählte Zahlenbeispiel war recht „günstig" – es verschwieg nämlich die erheblichen Rechenzeiten bei großen Zahlen. Die Zerlegung funktioniert wesentlich schneller, wenn man einige Rechenregeln für Primzahlen berücksichtigt.

Jede Primzahl paßt in die Form 6 n ± 1, wobei n eine ganze natürliche Zahl (außer 2 oder 3) darstellt. Das bedeutet aber nicht, daß jede aus dieser Formel resultierende Zahl auch eine Primzahl ist. Man braucht also nicht jede einzelne Zahl (1, 2, 3, 4, 5 usw.) oder alle ungeraden Zahlen daraufhin zu untersuchen, ob es sich um eine Primzahl handelt, sondern man rückt sinnvollerweise abwechselnd in Schritten von 2 und 4 vor, wobei man mit 5 beginnt: 5, 7, 11, 13, 17, 19, 23 usw.

Ferner ist es nur erforderlich, die zu untersuchende Zahl bis zu ihrer Wurzel zu dividieren, um herauszufinden, ob sie eine Primzahl ist. Das Resultat ist eine bei großen Zahlen ganz erheblich verkürzte Rechenzeit. *J. Grabau*

Anm. d. Red.: Wir verzichten hier auf den Abdruck des entsprechenden Programms und überlassen es dem Leser, die hier angegebenen Rechenregeln in Software umzusetzen!

Potenzrechnung mit negativer Basis

Viele programmierbare Taschenrechner, so auch der TI-59, gestatten die Verwendung der Taste y^x nur dann, wenn y größer als Null ist.

Für einfache Berechnungen ist bei einem negativen y-Wert im allgemeinen genug Übersicht vorhanden. In umfangreichen Programmen mit Schleifen usw. fehlt die Überschaubarkeit in den allermeisten Fällen. Es wird daher vorgeschlagen, Zahlenwerte mit den relativ häufig vorkommenden Exponenten 3, 4, 5 und 6 grundsätzlich anders zu behandeln, wie am folgenden Beispiel mit dem Wert y = –5 gezeigt werden soll.

Bisherige Methode

-5^3: 5 ± y^x 3 = 125

falsch (Vorzeichen) und blinkende Anzeige

Zweckmäßigere Methode

-5^3: 5 ± x x^2 = –125
-5^4: 5 ± x^2 x^2 = 625
-5^5: 5 ± x x^2 x^2 = –3125
-5^6: 5 ± x^2 x x^2 = 15 625

Wie man sieht, bringt die zweite Methode exakte und nichtblinkende Ergebnisse. Dabei ist nur bei Berechnungen mit den Exponenten 5 und 6 ein Tastendruck mehr auszuführen. Der Exponent 4 ist nur der Vollständigkeit halber mit angeführt worden. Es ist anzunehmen, daß den vielen Benutzern der oben geschilderte Rechnungsgang nicht bekannt ist, zumal auch Texas Instruments in keiner Beschreibung darauf hinweist. *Th. u. K. Reichel*

Monitor-Befehle des MZ-80 K

In dem MZ-80-K-Testbericht in Heft 6/1980, Seite 109, waren die zur Verfügung stehenden Monitorprogramm-Befehle nur zum Teil angegeben. Insgesamt versteht der Monitor des Sharp-Computers folgende Befehle:

LOADXYZ	Laden eines Programms mit dem Namen XYZ (Name darf auch weggelassen werden)
GOTO$XXXX	Start eines Maschinenprogramms an der hexadezimalen Adresse XXXX (stets vierstellig)
SG	Nach Ausführen dieses Befehls entsteht bei jedem Tastendruck ein kurzer Ton
SS	Hebt den Befehl SG wieder auf
FD	Start eines Maschinenprogramms an der Adresse F000, falls deren Inhalt Null ist

Die Monitor-Startroutine legt für den MUSIC-Befehl, Tempo (4) und Tonlänge (5) fest. Ferner beinhaltet der Monitor auch ein SAVE-Programm, was aber voraussetzt, daß die Speicherzellen $1102...1107 mit Programmlänge, Programmumfang usw. geladen sind. Der Monitor ist auch für die Tastenabfrage zuständig; z.B. fragt eine Routine bei $001E ab, ob „Shift Break" gedrückt ist – wenn ja, wird das Zero-Flag der CPU gesetzt. Die Tastatur wird über ein PIA 8255 (Adressen $E000...E001) angesprochen. *Rainer Schäfer*

µC-Nachrichten

SGS-Ates (zur italienischen IRI-STET-Gruppe gehörig) und **Zilog** (ein Exxon-Unternehmen) unterzeichneten einen Vertrag, daß SGS den 16-bit-Mikroprozessor *Z-8000* als Second Source liefert. Der Vertrag schließt auch die Peripheriebausteine der Z-8000-Familie ein sowie den Einchip-Computer Z-8.

□

Durch eine Reihe von Preissenkungen versucht Motorola, den Einsatz optoelektronischer Leitungen für die Datenübertragung interessanter zu machen: **Lichtsender und Detektoren** wurden um **30...40 % billiger.** Die Bauelemente sind mit den AMP-Verbindern kompatibel.

Der Micro-Shop Bodensee in 7778 Markdorf hat den Deutschland-Vertrieb des **Personal Computers der belgischen Firma DAI** übernommen (s. Tabelle in Heft 8/1980). Der Computer ist einer der preiswertesten mit Farb-Video-Ausgang.

□

AMI hat einen neuen **Einchip-Mikrocomputer** herausgebracht: Der Typ S 2200 beinhaltet einen Achtkanal-8-bit-A/D- und D/A-Umsetzer, zwei Prioritäts-Interrupts, einen Stromausfall-Interrupt und eine Stromsparschaltung für den Standby-Betrieb. Das ROM auf dem Chip besitzt 2 KByte Kapazität und ist extern auf 8 KByte erweiterbar. Der Stückpreis soll in der 100 000-Staffelung unter 20 DM liegen.

Josef Frank

8-KByte-Speicherkarte

Die hier beschriebene statische RAM-Karte eignet sich für den direkten Anschluß an die Computer CBM, PET, TRS-80, KIM-1, AIM-65, Nascom, Elzet-80 und Kontron-ECB. Von letzterem wurde auch die Steckerbelegung übernommen. Mit einigen wenigen Lötbrücken läßt sich die Karte an das jeweilige System adaptieren.

Auf der Universal-Speicherplatine, deren Schaltung *Bild 1* zeigt, befinden sich 16 RAM-Chips 2114 (1 K x 4 bit), aufgeteilt in zwei Blöcke von je 4 KByte. Ein 4-bit-Demultiplexer 74154 decodiert die oberen vier Adressenleitungen A 12...15 und erzeugt daraus 16 Auswahlsignale, d. h. 16 4-KByte-Blöcke, von denen sich zwei beliebige (CSA und CSB) mit Lötbrücken zum zweiten Decoder-IC durchschalten lassen. Die beiden 4-KByte-Blöcke sind dadurch unabhängig voneinander über den gesamten 64-KByte-Bereich verschiebbar; an Pin 19 des 74154 muß dazu ein Low-Signal angelegt sein.

Der zweite Decoder, ein 74LS139, decodiert die Adressenleitungen A 10 und A 11 und liefert 2 x 4 Chipselect-Signale, die die 8-KByte-Speicher ansprechen.

Der Adressenbus ist auf den Leitungen A 0...9 mit Invertern vom Typ 7404 gepuffert, die hier als einfache Treiber wirken. Die Invertierung hat keinen Einfluß auf die Adressierung, da sie lediglich eine interne Umorganisation bewirkt, die den Anwender jedoch nicht zu interessieren braucht. Die höherwertigen sechs Adressenleitungen wurden nicht gepuffert, da sie ja ohnehin nur jeweils an einen IC-Eingang geführt sind.

Der Datenbus ist mit zwei Oktal-Bus-Leitungstreibern 74LS244 gepuffert, wobei jeweils zwei Treiber antiparallel geschaltet sind (Schreiben/Lesen). IC 7 (7400) verknüpft die Auswahlleitungen CSA und CSB sowie die

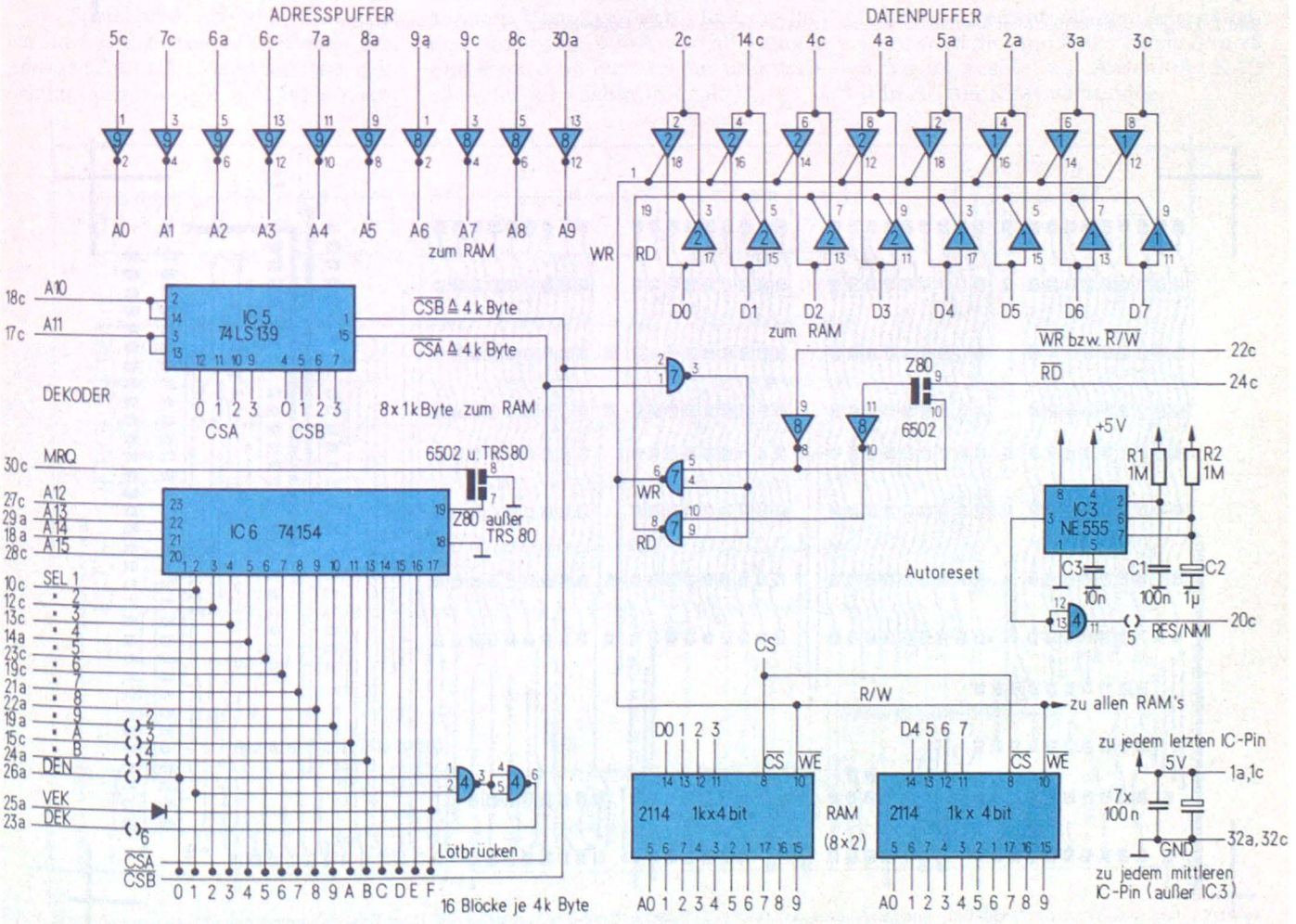


Bild 1. Vollständiges Schaltbild der 8-KByte-Speicherkarte. Die Lötbrücken dienen zur Adaption an unterschiedliche Mikrocomputersysteme

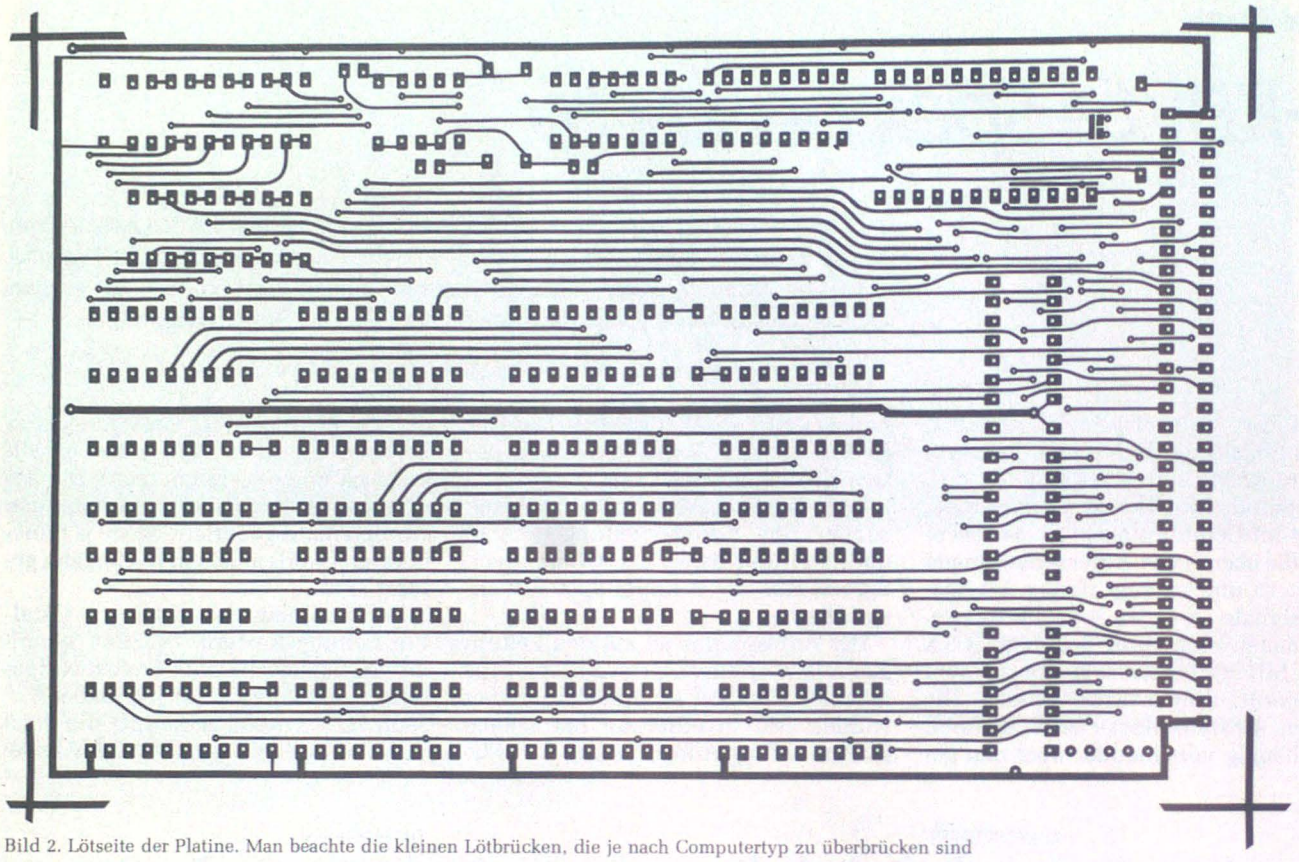


Bild 2. Lötseite der Platine. Man beachte die kleinen Lötbrücken, die je nach Computertyp zu überbrücken sind

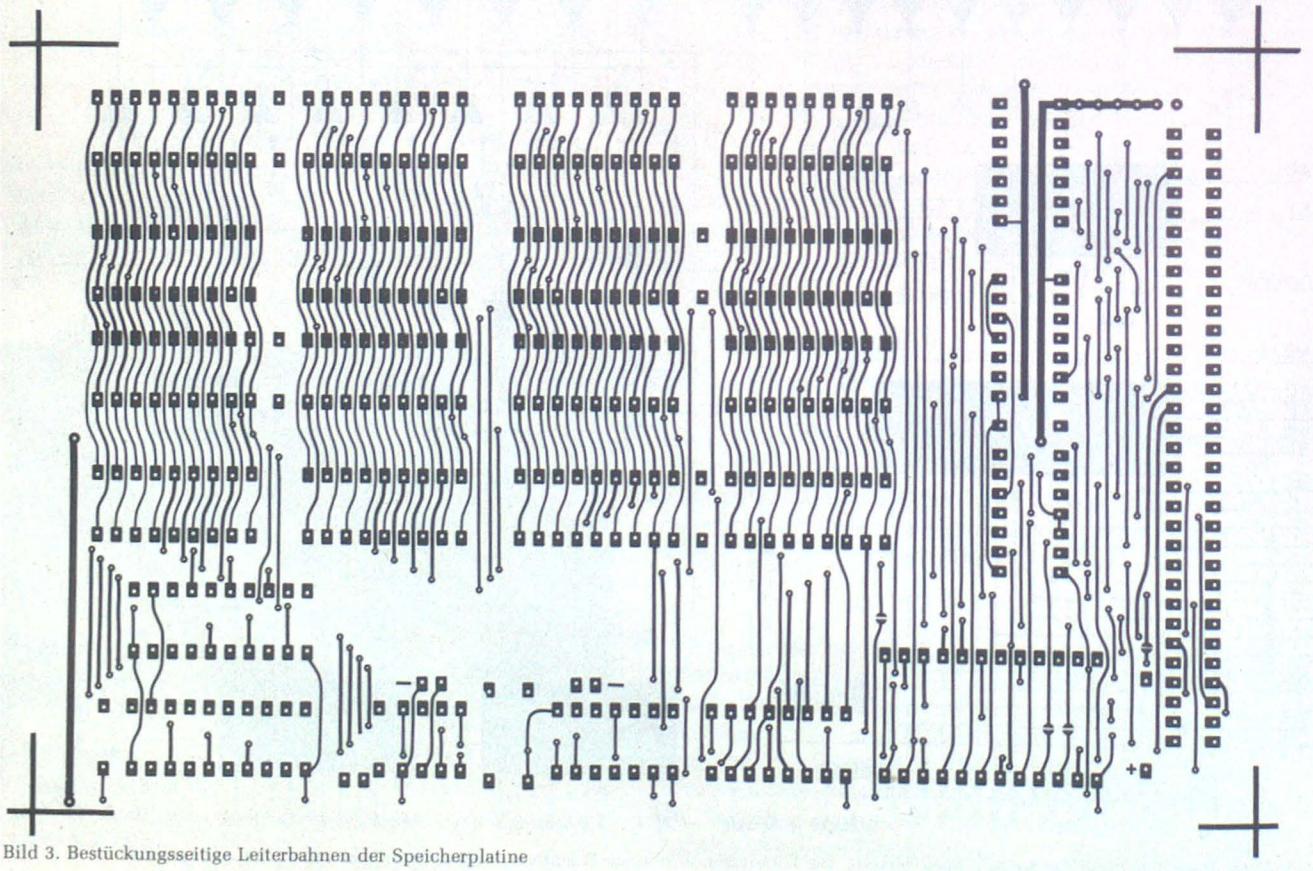


Bild 3. Bestückungsseitige Leiterbahnen der Speicherplatine

Schreib-Lese-Leitung R/W bzw. RD und WR (*Tabelle 1*).

Die Inverter 8/9, 8 und 8/11, 10 invertieren die WR-Leitung und die RD-Leitung (Z80), (Brücke 9 geschlossen), und steuern bei anliegendem Chipselect (CSA oder CSB) (Pin 7/3 auf High) mit den Gattern 7/4, 5, 6 und 7/10, 9, 8 die Datenpuffer. Bei 6502-Systemen fehlt die RD-Leitung, statt dessen wird die invertierte WR-Leitung (=R/W) verwendet (Brücke 10 geschlossen). Pin 7/6, dessen Low-Signal die Datenpuffer auf Schreiben schaltet, liefert gleichzeitig das WE-Signal (= write enable) für die RAM.

Die Datenbuffer schalten nur durch, wenn CSA oder CSB aktiv werden (Low-Signal). Wird eine der beiden Decoder-Lötbrücken weggelassen, z. B. CSA, so beansprucht die Karte nur 4 K Adreßbereich. Das zu CSA gehörende RAM bleibt ausgeblendet.

Bild 2 zeigt die Lötseite der Platine, *Bild 3* die bestückungsseitigen Leiterbahnen, und *Bild 4* den vollständigen Bestückungsplan.

Anschluß an den KIM-1

Der Speicherbereich des KIM-1 ist nur für 8 K decodiert: Die oberen drei Adressen A 13...A 15 sind nicht berücksichtigt, können also beliebige Werte annehmen. Dadurch kann dieselbe Speicherstelle auf 8 verschiedenen Adressen angesprochen werden: 0012-2012-4012 usw. Der Chip-Enable-Eingang des Decoders (decode enable, Pin A-K am KIM) wird nämlich nicht nur auf den ersten 8 K angesprochen, sondern vom Anwender über eine Drahtbrücke fest auf Masse gelegt. Werden nun zusätzlich 8-KByte Speicherplatz angeschlossen, so müssen natürlich auch die oberen Adressen für die Decodierung mit herangezogen werden. Der Speicherplatz auf der KIM-Platine (RAM, ROM u. Ports) darf dann nur noch auf den untersten 8 K (0000-1FFF) angesprochen werden können. Dazu „verodert" IC 4 die beiden untersten 4-K-Auswahlleitungen und erzeugt an Pin 23 a das erforderliche „decode enable"-Signal.

Tabelle 1: Pegel an den Auswahlleitungen

CSA	CSB	R/W	RD	WR	Pin 1 schreiben	Pin 19 lesen	Puffer Tristate
H	H	X	X	X	H	H	Karte nicht angesprochen
H	L	H	L	H	H	L	
L	H	H	L	H	H	L	lesen
H	L	L	H	L	L	H	
L	H	L	H	L	L	H	schreiben

Die 6502-CPU holt sich die Startadressen (Vektoren) für MMI, Reset und IRQ von den Speicherstellen FFFA-FFFF. Niedergelegt sind diese Vektoren aber im Monitor-ROM auf 1FFA-1FFF, wo sie durch die unvollständige Decodierung von der CPU gelesen werden konnten. Durch die jetzt vollständige Decodierung „findet" die CPU bei FFFA-FFFF aber nichts mehr. Der Rechner läßt sich dadurch mit einem Reset nicht mehr ins Monitor-Programm zurückholen. Die Lösung ist eine Diode, die den obersten 4-K-Block des Speicherbereichs, aus dem die Start-Vektoren geholt werden, zur oberen Hälfte des Monitor-ROMs, das diese Start-Vektoren enthält, parallelschaltet. Dazu wird die Anode der Diode (Pin 25 a, Vector enable) mit Anschluß „K-7" (Pin A-J am KIM) verbunden.

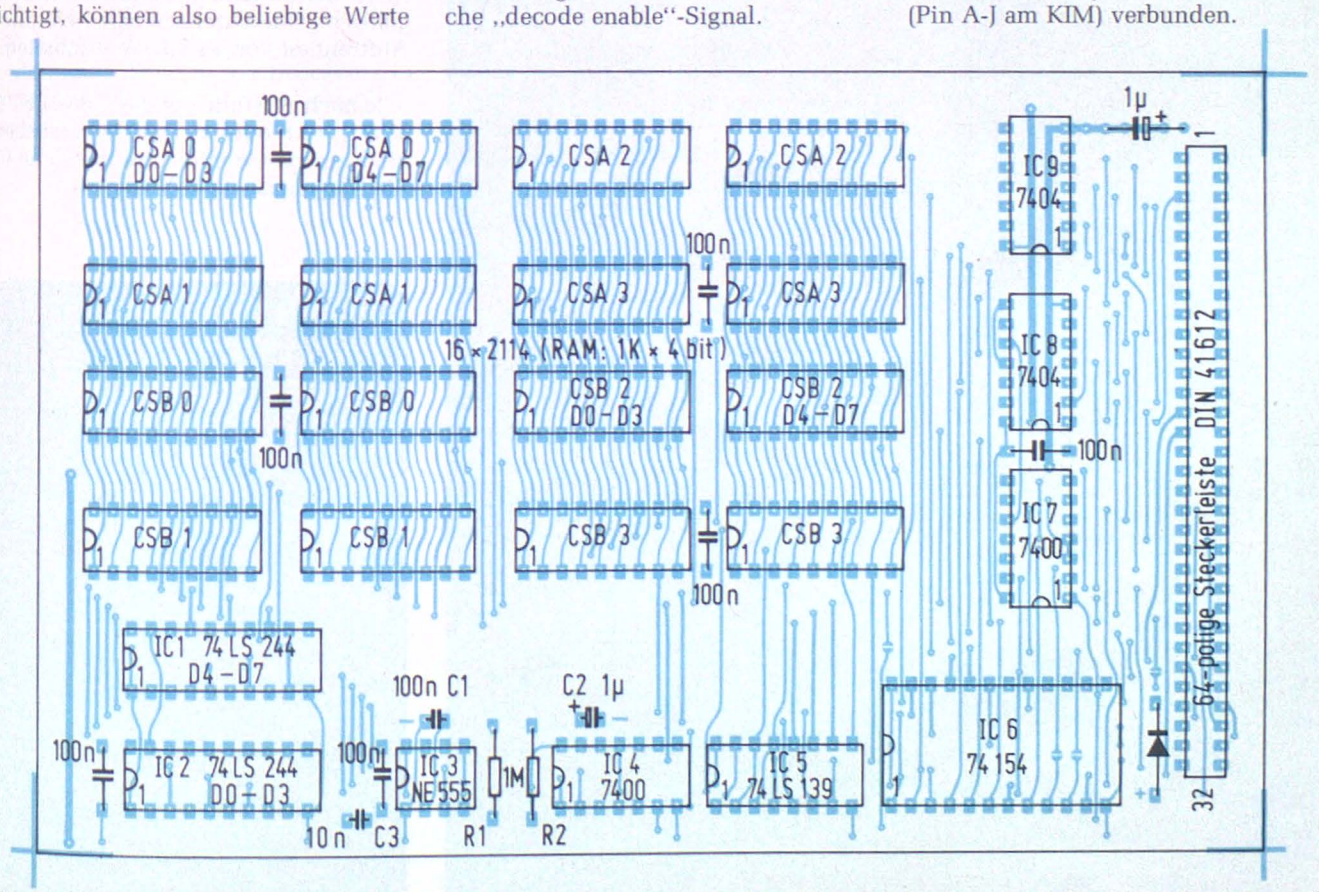


Bild 4. Bestückungsplan. Bei den Computern PET und CBM kann der Decoder-Baustein 74154 entfallen; seine Ausgänge sind direkt mit dem Select-Leitungen des Computers zu verbinden

Adressen- und Datenbus, R/W, Decode enable und Vector enable sind gemäß *Tabelle 2* mit der KIM-1-Platine zu verbinden. Es werden nur die Lötbrücken 6, 8 und 10 geschlossen. Am KIM ist die Drahtbrücke von Anschluß K des Application Connector nach Masse zu entfernen.

Die untersten beiden 4-KByte-Blöcke sind für die KIM-1-Platine reserviert, der oberste wird für die Reset-Vektoren benötigt. Der mögliche Adressenbereich ist also 2000...EFFF. Empfohlen wird die Verwendung des Bereiches 2000...3FFF, um Kompatibilität mit anderen Anwendern herzustellen.

Tabelle 2: Steckerbelegung und Anschlüsse

	ECB, ELZET 80	PET-CBM	KIM/AIM	TRS-80	NASCOM
A 0	5 c	B 1	E – A	25	21
A 1	7 c	B 2	E – B	27	23
A 2	6 a	B 3	E – C	40	24
A 3	6 c	B 4	E – D	34	22
A 4	7 a	B 5	E – E	31	19
A 5	8 a	B 6	E – F	35	18
A 6	9 a	B 7	E – H	38	17
A 7	9 c	B 8	E – J	36	16
A 8	8 c	B 9	E – K	11	15
A 9	30 a	B 10	E – L	17	12
A 10	18 c	B 11	E – M	4	14
A 11	17 c	B 12	E – N	9	13
A 12	27 c	—	E – P	5	9
A 13	29 a	—	E – R	6	10
A 14	18 a	—	E – S	10	11
A 15	28 c	—	E – T	7	20
D 0	2 c	B 33	E – 15	30	14
D 1	14 c	B 34	E – 14	22	15
D 2	4 c	B 35	E – 13	32	12
D 3	4 a	B 36	E – 12	26	8
D 4	5 a	B 37	E – 11	18	7
D 5	2 a	B 38	E – 10	28	9
D 6	3 a	B 39	E – 9	24	10
D 7	3 c	B 40	E – 8	20	13
D 8 (SEL1)	10 c	B 16	—	—	—
D 9 (SEL2)	12 c	B 17	—	—	—
D 10 (SEL3)	13 C	B 18	—	—	—
D 11(SEL4)	14 a	B 19	—	—	—
D 12(SEL5)	23 c	B 20	—	—	—
D 13 (SEL6)	19 c	B 21	—	—	—
D 14 (SEL7)	21 a	B 22	—	—	—
D 15 (SEL8)	22 a	—	—	—	—
+ 15 V (SEL 9)	19 a x	B 23	—	—	—
– 15 V (SEL A)	15 c x	B 24	—	—	—
VC MOS (SEL B)	24 a x	B 25	—	—	—
RD	24 c	—	—	15	25
WR (R/W)	22 c	B 30	E – 7	13	26
NMI(RES)3)x	20 c	B 27	J 3 – 7 1)	—	—
MRQ D P R	30 c	—	—	—	27
(DEK) x	23 a	—	A – K 2)	—	—
n0 (VEK) x WR.En	25 a	—	A – J 2)	—	—
(DEN) x	26 a	—	—	—	40

x = durch Lötbrücke isolierbar
DEK = decode enable KIM-1 (Drahtbrücke am KIM entfernen.)
VEK = vector enable KIM-1
DEN = decode enable für Nascom
1) = nur AIM-65
2) = nur KIM-1
3) = muß nicht angeschlossen werden; dann aber Karte nicht nach dem Computer einschalten, sie wird sonst nicht akzeptiert!

Anschluß an PET und CBM

Die Adressenleitungen von PET und CBM sind intern bereits voll decodiert. Die zehn Select-Leitungen am Speicher-Erweiterungsstecker sprechen direkt 4-K-Blöcke an und ersetzen dadurch den Decoder 74154. Dieser wird deshalb nicht bestückt; seine Ausgänge werden stattdessen mit den Select-Leitungen von PET/CBM verbunden.

Wird die Versorgungsspannung für die Speicherplatte erst nach dem Computer eingeschaltet, so erkennt Basic den zusätzlichen Speicherplatz nicht. Dazu muß erst ein Reset ausgegeben werden, der die CPU den gesamten Adressenbereich überprüfen läßt. Die gefundenen RAM-Bereiche werden dabei notiert und stehen dann dem CBM-Basic zur Verfügung. Dieser Reset wird auf der Speicherplatte beim Einschalten von einer Auto-Reset-Schaltung selbsttätig erzeugt, und liegt an Pin 20 c (zu verbinden mit B 27 am CBM). Im RAM befindliche Programme werden gelöscht. Der Autoreset läßt sich durch Auftrennen von Brücke 5 abschalten.

Je nach Ausführung des CBM/PET (4 K, 8 K, 16 K oder 32 K Speicherausbau) sind nur bestimmte Adreßbereiche für die Speicherkarte zulässig:

Tabelle 3: Stückliste

16 St.	2114
2 St.	7404
2 St.	7400
1 St.	NE 555
1 St.	74LS139 (74139)
1 St.	74 154
2 St.	74LS244
8 St.	100 nF
1 St.	10 nF
2 St.	1 µF/10 V (2,2 µF)
1 St.	1 N 4148
2 St.	1 MΩ
1,12 m	IC-Kontaktleisten
32 St.	9er IC-Isolatoren
4 St.	10er IC-Isolatoren
3 St.	8er IC-Isolatoren
8 St.	7er Isolatoren
2 St.	12er Isolatoren
1 St.	Platine (durchkont.)

PET/CBM-Version	Speicherkarte	Decodierung
4 K	1000-1FFF und 2000-2FFF	SEL 1 u. SEL 2
8 K	2000-2FFF und 3000-3FFF	SEL 2 u. SEL 3
16 K	4000-4FFF und 5000-5FFF	SEL 4 u. SEL 5
32 K	voll ausgebaut	

Es werden nur die Lötbrücken 2, 3, 4, 5, 8, 10 geschlossen. Die Diode bleibt unbestückt.

Anschluß an den TRS-80

Der TRS-80 ist wie der PET voll decodiert, hat aber keine herausgeführten 4-K-Select-Leitungen, sondern statt dessen den kompl. Adreßbus. Die Speicher-Karte muß die oberen 4 Adreßleitungen, wie auch bei allen anderen Mikrocomputern decodieren. Die Speicherplatte darf nicht nach dem Computer eingeschaltet werden, sie wird sonst vom TRS-80-Basic nicht akzeptiert. Die Speicherplatte muß in ihrem Adreßbereich unmittelbar an das intern bestückte RAM anschließen.

TRS-80-Version	Speicherkarte
4 K	5000-5FFF
8 K	6000-6FFF
16 K	8000-8FFF

Es werden nur die Lötbrücken 8 und 9 geschlossen, Diode und Autoreset bleiben unbestückt.

Die Fa. JFE (Wasserburger Landstr. 120, 8000 München 82) liefert alle erforderlichen Teile lt. der Stückliste in *Tabelle 3* einzeln oder als Komplettbausatz.

µC-Nachrichten

Die Computershop-GmbH (6236 Eschborn) bietet für den Mikrocomputer Sorcerer alias Sokrates Minidisketten-Stationen an, die mit einem CP/M-Betriebssystem und Basic-80 ausgeliefert werden. Es ist auch eine Version mit Doppellaufwerk und eingebautem 12-Zoll-Monitor mit 20 MHz Videobandbreite zu haben.

□

Vor den Toren Frankfurts, nämlich in Sprendlingen, hat Heathkit ein Demonstrationszentrum für seine Mikrocomputer nebst Peripherie (H-8, H-11, H-89, ET-3400, H-14 usw.) eingerichtet. Heathkit gehört übrigens seit einiger Zeit nicht mehr zum Schlumberger-Konzern, sondern wurde von der US-Firma Zenith übernommen.

□

In den USA wurde für den Apple-II ein neuer, preiswerter Drucker vorgestellt. Der „Silentype" ist sehr leise und druckt bis zu 80 Zeichen pro Zeile auf 8,5 Zoll breitem Thermopapier. Zeichen und Grafiken werden als 5x7-Punktmatrix erzeugt. Druckbreite, Zeilenabstand, Druckintensität u. a. können mit Steuerzeichen vom Computer aus fernbedient werden. Preis des Druckers in den USA: knapp 600 Dollar.

□

Hewlett-Packard entwickelte einen handlichen kleinen Lesestift für den Strichcode (Bar Code Reader) für eine minimale Linienbreite von ca. 0,3 mm. Die notwendige Decodierelektronik ist allerdings nicht eingebaut, am Ausgang stehen lediglich digitale Impulse an. Der Stift enthält eine Fotodiode, einen Verstärker und einen Schnitt-Trigger mit offenem Kollektor. Der Preis liegt allerdings bei über 200 DM.

□

Bisher gab es in Europa keine Organisation, die der amerikanischen CP/M Users Group entsprach. Inzwischen wurde aber ein European CP/M Users Club gegründet. Seine Aufgaben und Ziele sind der kostenlose Programmaustausch, die Übersetzung anderssprachiger Programme, die Information der Mitglieder über neue Software- und Hardware-Entwicklungen sowie der Vertrieb von deutschen und französischen CP/M-Betriebssystemen. Der Mitgliedsbeitrag von 80 Schweizer Franken pro Jahr dient zur Spesendekkung. Die Adresse lautet: Hadlaubstraße 56, CH-8006 Zürich, Schweiz.

Hubert Holewa

TRS-80-Disassembler

Der hier beschriebene Disassembler soll das Programmieren des TRS-80 in Maschinensprache erleichtern. Das Programm belegt die Speicherplätze von 6000H bis 6864H und kann an der Adresse 6000H gestartet werden.

Das Programm meldet sich nach dem Start mit folgendem Text: TRS-80 DISASSEMBLER 1.2

Das Asterisk-Zeichen (*) zeigt, daß sich der Disassembler im Befehlsmode befindet. Nun kann einer der folgenden drei Befehle eingegeben werden:

D = Disassemble (Disassemblierter Speicherbereich wird auf dem Bildschirm angezeigt)

P = Print (Adresse und dazugehöriger Speicherinhalt werden in hexadezimaler Form auf dem Bildschirm angezeigt)

J = Jump (Sprung zur angegebenen hexadezimalen Adresse)

Die Befehle ‚D' und ‚P' erfordern eine Anfangsadresse und eine Endadresse in der Form DXXXX YYYY, wobei XXXX die vierstellige hexadezimale Anfangsadresse und YYYY die vierstellige hexadezimale Endadresse ist. Der Befehl ‚J' benötigt nur eine vierstellige hexadezimale Sprungadresse, zu der das Programm dann springt (z.B. 1A19H = Rücksprung zu Basic).

Die Befehle mit den dazugehörigen Adressen werden mit nur einem Zwischenraum zwischen der Anfangsadresse und der Endadresse eingegeben. Während der Eingabe im Befehlsmode bleiben alle Cursorfunktionen wie Backspace und Space erhalten. Ein im Befehlsmode eingegebener Befehl wird mit der Taste „Enter" abgeschlossen. Hier ein Beispiel für die Eingabe eines Befehls:

***D6000 600C**

6000	310060	LD	SP,6000
6003	CDC901	CALL	01C9
6006	210F65	LD	HL,650F
6009	AF	XOR	A
600A	320B65	LD	(650B),A

Wird die Taste „Break" während der Ausführung eines der zwei Befehle „D" oder „P" gedrückt, wird die laufende Befehlsausführung bis zur Betätigung der Taste „Space" angehalten.

Soll ein unzulässiger OP-Code disassembliert werden, wird die Fehlermeldung „ERROR" neben dem unzulässigen OP-Code angezeigt. Nach einer Fehlermeldung befindet sich das Programm wieder im Befehlsmode.

Dieser Disassembler wurde zwar für den TRS-80 geschrieben, er ist jedoch nach Umänderung der Eingaberoutine und der Printroutine für jeden anderen Z-80-Mikrocomputer verwendbar. Das Programm kann nach Umänderung der absoluten Adressen in jeden anderen Speicherbereich übertragen werden. Da das Programm etwa 2K an Speicherplatz benötigt, ist es auch in einem TRS-80 mit 4K verwendbar.

```
**HEXDUMP TRS-80 DISASSEMBLE
6000 31 00 60 CD C9 01 21 0F 65 AF 32 0B 65 32 0C 65
6010 32 0E 65 CD DB 64 E5 D5 C5 AF 32 0A 65 3E 2A CD
6020 2C 64 21 FF 64 06 0B CD 40 00 38 ED C3 82 63 C1
6030 D1 E1 3A 0B 65 B7 C2 91 64 AF 32 0C 65 3A 0A 65
6040 B7 28 05 FD 23 CD AA 64 E5 D5 DD E5 D1 13 FD E5
6050 E1 ED 52 FA 5B 60 D1 E1 C3 16 60 FD E5 E1 7C CD
6060 B2 64 7D CD B2 64 CD 9D 64 3A 40 38 FE 04 20 07
6070 3A 40 38 FE 80 20 F9 D1 E1 FD 7E 00 CD B2 64 3A
6080 0A 65 B7 28 02 18 AB FD 7E 00 CD BF 60 CD 9D 64
6090 CD 9D 64 FE 76 CA 3A 61 FE DD 28 2C FE ED CA F8
60A0 63 FE FD 28 23 FE CB CA FF 62 FD 7E 00 E6 C0 CA
60B0 49 61 D6 40 CA EC 61 D6 40 CA 23 62 C3 49 62 F5
60C0 3A 20 40 32 0D 65 F1 C9 FE FD 20 04 3E 02 18 02
60D0 3E 01 32 0B 65 32 0C 65 FD 23 FD 7E 00 CD E2 60
60E0 18 C3 F5 C5 47 3A 20 40 4F 3A 0D 65 32 20 40 78
60F0 CD B2 64 CD BF 60 79 32 20 40 C1 F1 C9 F5 3A 0C
6100 65 B7 F2 11 61 FD 2B FD 7E 00 CD B2 64 FD 23 F1
6110 C9 FD 7E 01 CD E2 60 CD B2 64 FD 23 F1 C9 F5 FD
6120 7E 01 CD E2 60 FD 7E 02 CD E2 60 CD B2 64 FD 7E
6130 01 CD B2 64 FD 23 FD 23 F1 C9 E5 21 27 65 3E 00
6140 CD DB 64 E1 FD 23 C3 32 60 C5 E5 21 4F 65 FD 7E
6150 00 47 E6 07 CA DD 61 FE 07 CA DD 61 78 0E 00 E6
6160 08 28 02 0C 0C 0C 78 E6 0F 91 CD DB 64 21 2C 65
6170 78 E6 38 4F 78 E6 04 28 29 79 CB 3F CB 3F CB 3F
6180 FE 0A 20 02 4F 79 CD DB 64 78 E6 02 20 0A CD AA
6190 64 E1 C1 FD 23 C3 32 60 3E 2C CD 2C 64 CD FD 60
61A0 18 EC 78 E6 01 28 25 3A 0C 65 32 0B 65 79 CB 3F
61B0 CB 3F CB 3F CB 3F C6 08 CD DB 64 78 E6 0F FE 01
61C0 20 CC 3E 2C CD 2C 64 CD 1E 61 18 C2 78 21 7C 65
61D0 E6 38 CB 3F CB 3F CB 3F CD DB 64 18 B1 21 A8 65
61E0 78 E6 3C CB 3F CB 3F CD DB 64 18 A2 C5 E5 21 2C
61F0 65 3E 4C CD 2C 64 3E 44 CD 2C 64 CD 9D 64 FD 7E
6200 00 E6 38 CB 3F CB 3F CB 3F CD DB 64 3E 2C CD 2C
6210 64 FD 7E 00 E6 07 CD DB 64 CD AA 64 E1 C1 FD 23
6220 C3 32 60 E5 21 F2 65 FD 7E 00 E6 38 CB 3F CB 3F
6230 CB 3F CD DB 64 FD 7E 00 E6 07 21 2C 65 CD DB 64
6240 CD AA 64 E1 FD 23 C3 32 60 C5 E5 01 00 00 21 47
6250 66 FD 7E 00 FE CD 20 04 3E 03 18 3C E6 0F FE 03
6260 20 02 06 01 FE 06 20 02 06 02 FE 09 20 02 06 03
6270 FE 0B 20 02 06 04 FE 0E 20 02 06 05 78 B7 28 25
6280 05 FD 7E 00 E6 30 CB 3F CB 3F CB 3F CB 3F 4F 0C
6290 78 0D 28 04 C6 05 18 F9 CD DB 64 CD AA 64 E1 C1
62A0 FD 23 C3 32 60 21 16 66 FD 7E 00 47 E6 0F CD DB
62B0 64 CD 9D 64 78 E6 01 20 1E 21 3B 66 78 E6 38 CB
62C0 3F CB 3F CB 3F CD DB 64 78 E6 06 28 CE 3E 2C CD
62D0 2C 64 CD 1E 61 18 C4 78 3C E6 07 20 08 78 E6 38
62E0 CD B2 64 18 B6 21 2C 65 78 E6 30 CB 3F CB 3F CB
62F0 3F CB 3F FE 03 20 01 3C C6 08 CD DB 64 18 9C FD
6300 23 FD 7E 00 CD E2 60 3A 0B 65 B7 28 0D C6 80 32
6310 0C 65 FD 23 FD 7E 00 CD E2 60 C5 E5 21 B1 66 FD
6320 7E 00 47 E6 C0 20 26 78 E6 38 CB 3F CB 3F CB 3F
6330 CD DB 64 CD 9D 64 78 E6 07 21 2C 65 CD DB 64 CD
6340 AA 64 FD 23 E1 C1 AF 32 0C 65 C3 32 60 78 E6 C0
6350 CB 3F CB 3F CB 3F CB 3F CB 3F C6 07 CD DB
6360 64 CD 9D 64 78 E6 38 CB 3F CB 3F CB 3F C6 30 CD
6370 2C 64 3E 2C CD 2C 64 78 E6 07 21 2C 65 CD DB 64
6380 18 BD 21 FF 64 7E B7 FE 4A 28 14 FE 50 28 1D FE
6390 44 28 20 C3 19 60 23 7E B7 FE 20 20 01 23 C9 CD
63A0 96 63 CD D2 63 ED 53 AA 63 C3 19 1A 3A 0A 65 3C
63B0 32 0A 65 CD 96 63 CD D2 63 D5 FD E1 CD 96 63 CD
63C0 D2 63 D5 DD E1 C3 2F 60 7E FE 40 38 02 D6 07 D6
63D0 30 C9 CD C8 63 CB 27 CB 27 CB 27 CB 27 57 23 CD
63E0 C8 63 82 57 23 CD C8 63 CB 27 CB 27 CB 27 CB 27
63F0 5F 23 CD C8 63 83 5F C9 E5 C5 D5 FD 23 FD 7E 00
6400 CD E2 60 0E 00 11 D5 66 FD 7E 00 47 1A B8 28 0A
6410 0C 13 79 FE 36 D2 91 64 18 F2 79 21 0B 67 CD DB
6420 64 CD AA 64 D1 C1 E1 FD 23 C3 32 60 E6 7F FE 28
6430 20 09 F5 3A 0E 65 3C 32 0E 65 F1 FE 29 20 09 F5
6440 3A 0E 65 3D 32 0E 65 F1 FE 24 20 1E E5 21 3F 65
6450 3A 0E 65 B7 28 08 3A 0B 65 C6 03 32 0B 65 3A 0B
6460 65 CD DB 64 AF 32 0B 65 E1 C9 FE 25 20 04 CD FD
6470 60 C9 FE 21 CA 91 64 FE 2F 20 04 CD 9D 64 C9 FE
6480 23 20 04 CD 1E 61 C9 D5 FD E5 CD 33 00 FD E1 D1
6490 C9 E5 21 CE 66 AF CD DB 64 E1 C3 16 60 F5 3A 20
64A0 40 E6 F8 C6 08 32 20 40 F1 C9 F5 3E 0D CD 2C 64
64B0 F1 C9 D5 F5 57 CB 3F CB 3F CB 3F CB 3F E6 0F FE
64C0 0A 38 02 C6 07 C6 30 CD 2C 64 7A E6 0F FE 0A 38
64D0 02 C6 07 C6 30 CD 2C 64 F1 D1 C9 C5 E5 0E 00 47
64E0 7E E6 80 20 03 23 18 F8 78 B9 28 04 0C 23 18 F0
64F0 7E E6 7F CD 2C 64 23 7E E6 80 28 F4 E1 C1 C9 4A
6500 31 41 31 39 0D 30 31 30 30 0D 00 00 00 CE 00 D4
6510 52 53 2D 38 30 20 44 49 53 41 53 53 45 4D 42 4C
6520 45 52 20 31 2E 32 0D C8 41 4C 54 0D C2 C3 C4 C5
6530 C8 CC A8 24 29 C1 C2 43 C4 45 A4 D3 50 C1 46 C8
6540 4C C9 58 C9 59 C8 4C C9 58 2B 25 C9 59 2B 25 CC
6550 44 2F CC 44 2F C9 4E 43 2F C9 4E 43 2F C4 45 43
6560 2F CC 44 2F C1 44 44 2F 24 2C CC 44 2F C4 45 43
6570 2F C9 4E 43 2F C4 45 43 2F CC 44 2F A8 42 43 29
6580 2C 41 C1 2C 28 42 43 29 A8 44 45 29 2C 41 C1 2C
6590 28 44 45 29 A8 23 29 2C 24 A4 2C 28 23 29 A8 23
65A0 29 2C 41 C1 2C 28 23 29 CE 4F 50 D2 4C 43 41 C5
65B0 58 2F 41 46 2C 41 46 27 D2 52 43 41 C4 4A 4E 5A
65C0 2F 25 D2 4C 41 CA 52 2F 25 D2 52 41 CA 52 2F 4E
65D0 5A 2C 25 C4 41 41 CA 52 2F 5A 2C 25 C3 50 4C CA
65E0 52 2F 4E 43 2C 25 D3 43 46 CA 52 2F 43 2C 25 C3
65F0 43 46 C1 44 44 2F 41 2C C1 44 43 2F 41 2C D3 55
6600 42 2F D3 42 43 2F 41 2C C1 4E 44 2F D8 4F 52 2F
6610 CF 52 2F C3 50 2F D2 45 54 D0 4F 50 CA 50 A1 C3
6620 41 4C 4C D0 55 53 48 A1 D2 53 54 D2 45 54 A1 CA
6630 50 A1 C3 41 4C 4C A1 A1 D2 53 54 CE 5A DA CE 43
6640 C3 D0 4F D0 45 D0 CD CA 50 2F 23 C1 44 44 2F 41
6650 2C 25 D2 45 54 C3 41 4C 4C 2F 23 C1 44 43 2F 41
6660 2C 25 CF 55 54 2F 25 2C 41 D3 55 42 2F 25 C5 58
6670 58 C9 4E 2F 41 2C 25 D3 42 43 2F 41 2C 25 C5 58
6680 2F 28 53 50 29 2C 24 C1 4E 44 2F 25 CA 50 2F 28
6690 24 29 C5 58 2F 44 45 2C 24 D8 4F 52 2F 25 C4 49
66A0 CF 52 2F 25 CC 44 2F 53 50 2C 24 C5 49 C3 50 2F
66B0 25 D2 4C 43 D2 52 43 D2 4C D2 52 D3 4C 41 D3 52
66C0 41 A1 D3 52 4C C2 49 54 D2 45 53 D3 45 54 C5 52
66D0 52 4F 52 0D A1 40 41 42 43 44 45 46 47 48 49 4A
66E0 4B 4D 50 51 52 53 56 57 58 59 5A 5B 5E 60 61 62
66F0 67 68 69 6A 6F 72 73 78 79 7A 7B A0 A1 A2 A3 A8
6700 A9 AA AB B0 B1 B2 B3 B8 B9 BA BB C9 4E 2F 42 2C
6710 28 43 29 CF 55 54 2F 28 43 29 2C 42 D3 42 43 2F
6720 48 4C 2C 42 43 CC 44 2F 28 23 29 2C 42 43 CE 45
6730 47 D2 45 54 4E C9 4D 2F 30 CC 44 2F 49 2C 41 C9
6740 4E 2F 43 2C 28 43 29 CF 55 54 2F 28 43 29 2C 43
6750 C1 44 43 2F 48 4C 2C 42 43 CC 44 2F 42 43 2C 28
6760 23 29 D2 45 54 49 C9 4E 2F 44 2C 28 43 29 CF 55
6770 54 2F 28 43 29 2C 44 D3 42 43 2F 48 4C 2C 44 45
6780 CC 44 2F 28 23 29 2C 44 45 C9 4D 2F 31 CC 44 2F
6790 41 2C 49 C9 4E 2F 45 2C 28 43 29 CF 55 54 2F 28
67A0 43 29 2C 45 C1 44 43 2F 48 4C 2C 44 45 CC 44 2F
67B0 44 45 2C 28 23 29 C9 4D 2F 32 C9 4E 2F 48 2C 28
67C0 43 29 CF 55 54 2F 28 43 29 2C 48 D3 42 43 2F 48
67D0 4C 2C 48 4C D2 52 44 C9 4E 2F 4C 2C 28 43 29 CF
67E0 55 54 2F 28 43 29 2C 4C C1 44 43 2F 48 4C 2C 48
67F0 4C D2 4C 44 D3 42 43 2F 48 4C 2C 53 50 CC 44 2F
6800 28 23 29 2C 53 50 C9 4E 2F 41 2C 28 43 29 CF 55
6810 54 2F 28 43 29 2C 41 C1 44 43 2F 48 4C 2C 53 50
6820 CC 44 2F 53 50 2C 28 23 29 CC 44 49 C3 50 49 C9
6830 4E 49 CF 55 54 49 CC 44 44 C3 50 44 C9 4E 44 CF
6840 55 54 44 CC 44 49 52 C3 50 49 52 C9 4E 49 52 CF
6850 54 49 52 CC 44 44 52 C3 50 44 52 C9 4E 44 52 CF
6860 54 44 52 A1 86 00 00 00 00 00 00 00 00 00 00 00
```

Peter Stuhlmüller

Superbrain: Zweifach-Prozessorsystem für Anspruchsvolle

Das Computersystem mit dem etwas überheblichen Namen „Superbrain" (Super-Gehirn) ist frühzeitig mit Vorschußlorbeeren bedacht worden. Vorschußlorbeeren sind schnell vergeben. Für den Superbrain sind sie jedoch nicht vergeudet worden. Dieser Bericht zeigt warum.

Schon auf den ersten Blick erweckt der „Superbrain" eine gehörige Portion Eindruck. Das schwungvoll geformte Gehäuse mit dem augenfälligen Übergang von der Tastatur auf die Bildschirmebene zeigt einen gut gelungenen Entwurf. Doch für den etwas nüchteren Charakter, den Computeristen nun mal haben, sind Äußerlichkeiten gewissermaßen „blendfrei", wenn das Innere nichts taugt. Der Superbrain hält jedoch auch im Inneren, was sein Äußeres verspricht.

„Aufregende" Hardware

Computeranwender blicken schon gerne mal auf das „Fahrgestell" ihrer Maschine, auf dem die große Software laufen soll. Der Superbrain hat hier einiges zu bieten. Das System enthält zwei CPUs des Typs Z 80A, von denen die eine ausschließlich für den Anwender bereit steht und die andere das lästige und oft zeitraubende, aber doch notwendige Handling mit der Peripherie abwickelt. Betrachtet man die sehr gut gegliederte Platine genauer, so traut man kaum seinen Augen: sitzen da doch 64 KByte dynamische RAMs (4116) unmittelbar neben der Anwender-CPU. Damit offenbart sich die Konzeption des Superbrain. Dieser Computer soll Programme höherer Sprachen verarbeiten können, die nicht auf beengtem Speicherplatz ein Provisorium darstellen – im Hinblick auf einen späteren Minicomputer-Einsatz. Weiterhin soll die Anwender-CPU im Programmlauf nicht durch das gleichzeitige Bearbeiten von Systemdienstprogrammen gestört werden. Die A-Version der CPU (4-MHz-Takt) sollte eine schnellere Bearbeitung weiter unterstützen.

Als weitere Besonderheit ist der CRT-Steuerbaustein 8350 in I^2L-Technologie zu nennen, der mit einer Punktfrequenz von 10,92 MHz arbeitet und für das CRT-Zeichenformat von 24 Zeilen (80 Zeichen pro Zeile) mit einem programmierbaren Punktraster ausgelegt ist. Der Bildschirm des Superbrain wird bei der Auflistung eines Programms so schnell beschrieben, daß es unmöglich ist, mit dem Auge beim Beschreiben einer Zeile zu folgen. Weiterhin ergibt dieser Baustein eine vielfältige Cursor-Programmbarkeit, eine direkte Zeichenspeicher-Adressierung von 4 K, 50/60 Hz Synchronisation für den CRT-Monitor, eine Programmier-

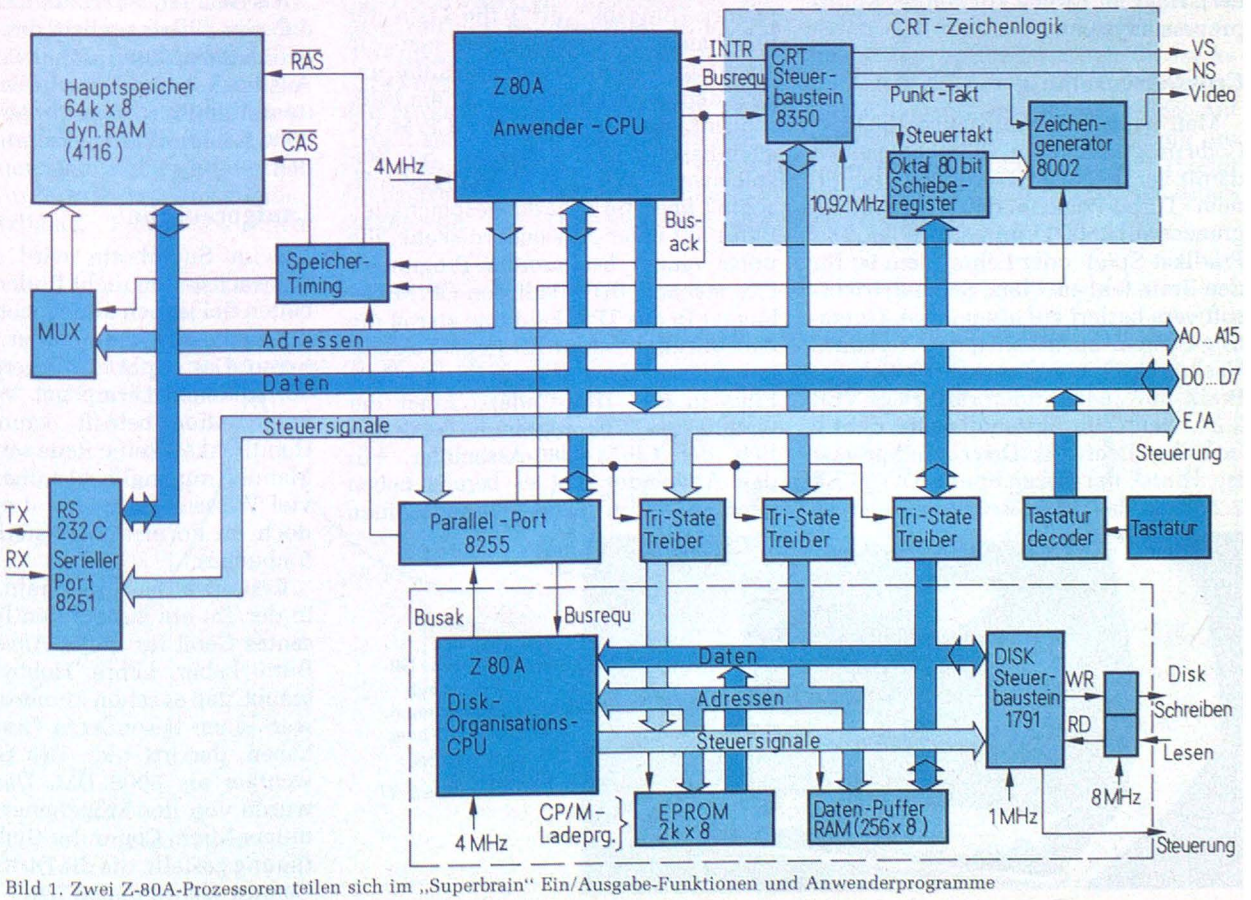


Bild 1. Zwei Z-80A-Prozessoren teilen sich im „Superbrain" Ein/Ausgabe-Funktionen und Anwenderprogramme

barkeit der Zeichen pro Zeile und der Zeichenfeldgröße (bis 16 x 16 Punkte).

Als Zeichengenerator wurde der maskenprogrammierte SMC 8002 ausgewählt, der über einen Vorrat von 128 Zeichen (Alphanumerik und Grafik) in einem 7 x 11 Raster verfügt. Neben breiter und schmaler Grafikdarstellung (alphanumerisch) ist auch eine schnelle Einzelpunktgrafik möglich. Die beiden Hochleistungs-CRT-Bausteine (8350 und 8002) ermöglichen den Aufbau einer für ihre Leistung bemerkenswert überschaubaren CRT-Zeichenlogik im Brain.

In einem Computer der Leistungsklasse eines Superbrain dürfen selbstverständlich Floppy-Laufwerke nicht fehlen. Der Brain hat zwei davon, die insgesamt eine Speicherkapazität von 320 KByte ergeben. Es handelt sich um $5^1/_4$"-Laufwerke, die einseitig, aber mit doppelter Speicherdichte softsektoriert betrieben werden. Die Datenformatierung übernimmt ein Western Digital Steuerbaustein, der FD 1791/2, der nach dem IBM System 34 formatiert (doppelte Speicherdichte).

Bild 1 zeigt das Blockdiagramm des Innenaufbaus. Die Zusammenarbeit der beiden Z-80A-CPUs, die über einen Parallel-Port 8255 eine Busverbindung besitzen (Daten- und Steuerleitungen, Adressen separat über Tri-State-Treiber), zeigt im Ansatz ein kleines Multiprozessorsystem.

Crossoftwarefähig

Man kann den Brain auch im Hex-Code programmieren. Dies wäre jedoch damit vergleichbar, würde man bei einem Turbo-Porsche den Drehzahlbegrenzer auf 2500 U/min einstellen. Das Prädikat Spiel- oder Lehrsystem ist für den Brain fehl am Platz. Seine Betriebssoftware basiert auf einem Disk-Operating-System amerikanischer Herkunft. Nach dem Einschalten meldet sich der Brain mit „64K Superbrain DOS VER 2.2", wenn die Systemdiskette richtig im linken Laufwerk (Drive A) eingesetzt ist. Hinter der Bezeichnung DOS VER 2.2 steckt das CP/M-Betriebssystem. Es bezeichnet das Monitor-Steuerprogramm für Mikrocomputer, das auf IBM-Format kompatiblen, flexiblen Disketten gespeichert ist. Das CP/M-Betriebssystem besteht aus einer Mehrzahl unabhängiger Software-Hilfsmittel, die ein effektives Arbeiten mit einem Computer der Art des Brain erst ermöglicht. In den Staaten ist es relativ weit verbreitet, in der alten Welt so gut wie gar nicht, leider. CP/M ist in vier Elemente logisch unterteilt:

BIOS behandelt allgemeines Handling mit der Systemperipherie (Disk-Laufwerke, TTY, CRT, Lochstreifen- und Bandleser, Drucker...)

BDOS liefert die Disk-Organisation für ein oder mehrere Laufwerke, und der damit verbundenen, aber unabhängigen File-Zuordnung.

CCP liest bestimmte File-Adressen aus den Programmbefehlen und initialisiert folgende Hilfsmittel wie Assembler, Editor, Debugger oder Linker/Relocator, die unter der Bezeichnung TPA abgelegt sind.

TPA bezeichnet Standard- und übergeordnete Hilfsprogramme für den Anwender.

CP/M ist eigentlich auf 8080-Systeme zurückzuführen. Läßt man sich über Directory (Eingabe „dir") die Liste der File-Namen ausgeben, findet man die angedeuteten Software-Hilfsmittel unter Kurzzeichen (ausgedruckt am Bildschirm), die auf der Brain CP/M-Systemdiskette enthalten ist. So beispielsweise 64 KTEST, STAT (gibt Aufschluß über die Zahl der gespeicherten Bytes auf einer Anwenderdiskette, die unter einem bestimmten Programm-File stehen), DDT (lädt den CP/M Debugger in das TPA-Feld und startet die Ausführung), ferner ED (lade Texteditor), DUMP (Umsetzen des Inhalts eines Files in den HEX-Code)... Einer der wichtigsten Hilfsprogramme ist sicherlich der CP/M-8080-Assembler. Für den Anwender gibt es bereits neben dem ASM (für 8080-CPU) auch einen Assembler mit dem Z-80-Source-Code (M 80) auf CP/M, jedoch nicht von Intertec.

Das CP/M-Betriebssystem bietet den Vorteil der Cross-Software-Fähigkeit. Jedes Programm, das mit CP/M auf Diskette geschrieben wurde, läuft auch auf irgendeiner anderen Maschine mit diesem Betriebssystem.

Die für viele Anwender sicher entscheidende Eigenschaft ist die Vorbereitung des Brain für das Arbeiten mit den wichtigsten Programmiersprachen Fortran (ANSI X3.9-1966), Cobol (ANSI 1974 Level 1 und Level 2 Elemente) und dem Basic 80 (Ausgabe 5.0), die alle aus dem Softwarehaus Microsoft stammen. Im Lieferumfang von Intertec Data Systems ist nur die CP/M-Systemdiskette enthalten. Die Compiler-Systemdisketten müssen separat erworben werden. Das Basic-80 ist identisch mit dem extended Disk-Basic für die Intellec Systeme MDS 800, bzw. MDS 888. Es beinhaltet 21 Befehle, 39 Statements und 48 Funktionen, eine Zeileneditier-Möglichkeit (CTRL U, Zeilennummer) und selbstverständlich die vollkommene Disk-I/O-Fähigkeit (sowohl sequentiell als auch im Speicherzugriff-Modus). Der Integer-Wertebereich liegt zwischen –32 768 und +32 767. Eine Zeilennummer kann bis zu 255 Zeichen einschließen.

Als Nachteil ist zweifellos erkennbar, daß die Überlegenheit des Z 80A gegenüber dem 8080 nicht zwingend zum Ausdruck kommt, weil alle drei Disk-Compiler (bzw. Interpreter) mit dem 8080-Source-Code geschrieben wurden.

Laufgeräusche

Beim Superbrain wird man echte Schwachstellen nicht finden. Beim Arbeiten fiel jedoch unangenehm auf, daß die Disk-Laufwerkmotoren ständig rotieren. Das ergibt einen geringen, aber vorhandenen Lärmpegel. Was die Dokumentation betrifft, kann hier von Handlichkeit keine Rede sein. Das User Manual (nur englisch) enthält zwar sehr viel Wissenswertes gut dargestellt, jedoch im Format eines stärkeren Telefonbuches.

Resümee: Der Superbrain (*Bild 2*) ist in der Tat ein außerordentlich interessantes Gerät für breite Anwendungen: Büro, Labor, Lehre, Hobby. Wer hier glaubt, daß es schon immer etwas teurer war, einen besonderen Geschmack zu haben, der irrt hier. Das Gerät kostet weniger als 8000 DM. Das Testgerät wurde von der Münchener Firma Heninger Micro Computer GmbH zur Verfügung gestellt, die die Distribution des Gerätes übernommen hat.

Bild 2.
Rechts sind die beiden Floppy-Laufwerke zu erkennen, die von BASF oder Shugart bezogen werden

Dieter Smode

Numerische Integration in Basic

Die Integration ist praktisch die Bestimmung einer Fläche, die von einer beliebigen mathematischen Funktion und der x-Achse des Koordinatensystems umgrenzt wird. Beispielsweise läßt sich damit der Effektivwert beliebiger Spannungsverläufe ausrechnen.

Das Programm verwendet die Simpson'sche Regel zur numerischen Lösung von bestimmten Integralen, deren Integrand formelmäßig vorliegt:

$$\int_a^b f(x)\,dx = \frac{h}{3}\,f(a) + f(b) + 4\sum_{k=0}^{N-1} f(a+(2k+1)h) + 2\sum_{k=1}^{N-1} f(a+2kh) + E(a, a+N\cdot 2h) \qquad (1)$$

wobei h die Schrittweite und N die (gerade) Schrittzahl ist. Dabei beschreibt folgende Beziehung den Fehler des Verfahrens:

$$E(a, a+N\cdot 2h) \approx -\frac{b-a}{180}\,h^4\,f^{(4)}(z), \quad z \in [a, b] \qquad (2)$$

$f^{(4)}$ = 4. Ableitung von f

In den Verfahrensfehler geht also die Schrittweite h mit der 4. Potenz ein, so daß der Fehler mit kleiner werdendem h sehr schnell kleiner wird.

Das eigentliche Programm *(Bild)* besteht aus zwei Teilen. Der erste Teil (Zeilen 100 bis 160) dient hauptsächlich zur Eingabe des Integranden. In einer INPUT-Anweisung holt sich der Rechner die vom Anwender gewünschte Funktion und gibt diese sofort wieder auf den Bildschirm aus, setzt jedoch „230DEFFNY (X) =" vor die Funktion.

Dann wird ausgenutzt, daß der Tastaturpuffer nur bei direkter Eingabe benötigt wird, während des Programmablaufes jedoch 10 Zeichen von der Tastatur empfangen kann. Per Programm wird der Tastaturpuffer nun wie folgt automatisch beschrieben:
3 Cursor hoch (145, der Cusor steht dann über der zuletzt vom Rechner ausgegebenen Zeile)
1 Return (13, die Zeile wird in den Programmspeicher übernommen)
GOTO 170 Ret. (71, 207, 49, 55, 48, 13, die Programmausführung wird ab Zeile 170 wieder aufgenommen)

Die oben angeführten Zahlen entsprechen der ASCII-Darstellung der erläuterten Befehle, wobei GOTO durch „G" und shifted „O" abgekürzt wurde. Mit POKE 525,10 wird dem Rechner dann mitgeteilt, daß sich 10 gültige Zeichen im Tastaturpuffer befinden. Die END-Anweisung bewirkt, daß der PET in den Direktmodus geht und die Zeichen aus dem Tastaturpuffer geholt, analysiert und in den Basis-Input-Puffer gebracht werden, wo sie dem Betriebssystem zur Bearbeitung zur Verfügung stehen.

Durch dieses Vorgehen wird nicht nur die gewünschte Zeile übernommen, sondern auch alle Basic-Variablen wieder initialisiert, so daß es keinen Zweck hat, einer im zweiten Programmteil benötigten Variablen im ersten Teil einen Wert zuzuweisen!

Der zweite Programmteil basiert auf den obigen Formeln und bedarf keiner besonderen Erläuterung. Um sicherzustellen, daß die vorzugebende Schrittzahl gerade ist, wird dies gleich nach

```
100 DATA145,145,145,13,71,207,49,55,48,13
110 POKE59468,14:PRINT"sQQQ
120 PRINTTAB(5)"R***NUMERISCHE INTEGRATION***r"
130 REM DIETER SMODE, 13-12-1978
140 PRINT"QQQDie zu integrierende Funktion ist nach":PRINT"dem '?' einzugeben."
150 INPUT"QQFNY(X)=";A$:PRINT"230DEFFNY(X)="A$
160 FORI=527TO536:READJ:POKEI,J:NEXT:POKE525,10:END
170 PRINT"s"
180 PRINT"Berechnung mit halber Schrittweite und":PRINT"anschliessende Fehler";
190 PRINT"schaetzung? (J/N)
200 GETA$:IFA$=""THEN200
210 IFA$<>"J"THEN230
220 IFA$="J"THENZ=1
230 DEFFNY(X)=2
240 PRINT:INPUT"Untere Grenze";U:INPUT"Obere Grenze";O
250 INPUT"QSchrittzahl";S:K=SAND1:IFK=0ANDS>=2THEN270
260 PRINT"QQDie Schrittzahl muss eine gerade ganze":PRINT"Zahl sein!!":GOTO250
270 H=(O-U)/S:PRINT"Schrittweite h=";H
280 F=FNY(U)+FNY(O)+4*FNY(U+H)
290 IFS=2THEN310
300 FORK=1TOS/2-1:F1=FNY(U+(2*K+1)*H):F2=FNY(U+2*K*H):F=F+4*F1+2*F2:NEXT
310 F=F*H/3:PRINT"QFlaecheninhalt=";F
320 PRINT:FORK=1TO40:PRINT"*";:NEXT:IFZ=0THEN180
330 IFZ=2THEN350
340 G=F:H=H/2:S=2*S:Z=2:GOTO270
350 F=(16*F-G)/15:PRINT"QFlaeche nach Fehlerschaetzung=":PRINTF:Z=0
360 PRINT:FORK=1TO40:PRINT"=";:NEXT
370 PRINT"Berechnung in den gleichen Grenzen mit"
380 PRINT"anderer Schrittweite?   (J/N)":PRINT
390 GETA$:IFA$=""THEN390
400 IFA$="J"THEN420
410 GOTO170
420 INPUT"Schrittzahl=";S
430 PRINT"Qmit Fehlerschaetzung? (J/N)"
440 GETA$:IFA$=""THEN440
450 IFA$="J"THENZ=1
460 PRINT:GOTO270
```

Dieses PET-Basic-Programm integriert beliebige Funktionen nach dem Simpson'schen Näherungsverfahren. Eine Besonderheit: Die Funktion kann vom Benutzer während des Programmablaufes eingetastet werden!

der Eingabe überprüft (K = S AND 1 ergibt nur dann Null, wenn S gerade ist!).

Da bei der numerischen Berechnung der Fehler des Ergebnisses nicht zu überblicken ist, führt das Programm – auf Wunsch – nach Beendigung des ersten Rechenganges einen zweiten Rechengang mit halber Schrittweite durch und berechnet aus beiden Ergebnissen mit Hilfe einer Fehlerabschätzung ein Endergebnis. Sollten die Ergebnisse der beiden Rechengänge stark differieren, so ist i. a. der Fehler noch recht groß; Abhilfe schafft dann eine Verkleinerung der Schrittweite (größere Schrittzahl).

Beispiel: $\int_0^{\ln 1000} e^x dx = 999$

Schrittzahl	Ergebnisse	rel. Fehler
2	1298.07211	29,9 %
	1035.04134	3,61 %
	1017.50596	1,85 %
12	999.586188	0,059 %
	999.037718	0,00378 %
	999.001153	0,000115 %

Das starke Differieren der beiden ersten Ergebnisse weist auf einen noch relativ großen Fehler hin, während das Endergebnis (Schrittzahl 12) schon ziemlich genau ist.

Allgemein gilt für die Fehlerabschätzung:

$$F = F_{h1} + \frac{F_{h1} - F_{h2}}{(h_2/h_1)^q - 1} \quad \text{mit } h_1 < h_2 \qquad (3)$$

wobei der Exponent q die Fehlerordnung des Verfahrens (bei Simpson q = 4) darstellt. Daraus folgt aber ($h_2 = 2\,h_1$):

$$F = \frac{16\,F_{h1} - F_{h2}}{15} \qquad (4)$$

Der zweite Teil des Programmes dürfte mit geringfügigen Änderungen auf jedem Basic-Rechner laufen. CBM-30XX-Besitzer und PET-Besitzer mit CBM-Betriebssystem müssen in Zeile 160 die Adressen für den Tastatur-Puffer (525, 527...536) ändern, so daß das komplette Programm übernommen werden kann.

Literatur

[1] *Jordan-Engeln, G.; Reutter, F.:* Numerische Mathematik für Ingenieure. BI-Hochschultaschenbuch 104

[2] *Jordan-Engeln, G.; Reutter, F.:* Formelsammlung zur numerischen Mathematik mit Fortran-IV-Programmen. BI-Hochschultaschenbuch 106

Stichworte zum Inhalt

Simpson'sche Regel, PET, CBM, Tastaturpuffer, DEFFN

Schnell-Sortierer

Wie hinlänglich bekannt, brauchen Sortiervorgänge in Basic ziemlich viel Zeit. Bei den bekannten Verfahren wie Bubble- oder Shellsort braucht man für 200 Daten schon mehr als 20 Minuten. Die Zeit für zusätzliche Daten steigt nicht linear, sondern logarithmisch an. Die hier vorgestellte Routine ist mehr als 100mal schneller (*Bild*). Bei 400 Daten benötigt die Routine nur 3,62 Sekunden! Ferner steigt die Sortierzeit mit der Datenmenge praktisch linear an.

Das abgedruckte Programm läuft unverändert auf PET und CBM. Nachdem ein Array dimensioniert und mit Zufallszahlen belegt wurde, beginnt der Sortiervorgang in Zeile 210. Dabei findet ein Transfer in ein zweites Array statt. Wenn die nur zur Demonstration vorhandene Zeitanzeige nicht gewünscht wird, können die Zeilen 200, 220 und 240 entfernt werden.

Die Vorteile der Methode sind ihre Geschwindigkeit, die einfache Programmierung (praktisch alles findet in Zeile 210 statt), das Erkennen von nicht belegten Plätzen im Array und die automatische Zusammenfassung von Doppelbelegungen. Nachteile sind, daß nur Zahlen, nicht aber Strings sortierbar sind, daß viel Arbeitsspeicher benötigt wird (zwei Felder!) und daß der Zahlenbereich relativ klein sein muß, da er gleichzeitig als Index dient. Selbstverständlich wäre auch ein Einlesen der Zahlen über DATA-Statements möglich.

D. Nussbaumer

Kassetten-Magazin

Für Besitzer der Mikrocomputer PET und CBM brachte die Firma Syntax (Rastatt) ein Kassetten-Magazin heraus, das im Abonnement bezogen werden kann: Monatlich erscheint eine Kassette mit fünf Basic-Programmen für 15 DM. Die erste solche Kassette enthielt einen Reaktionstest für Autofahrer, ein Nimm-Spiel, ein Menü-Programm zum schnellen Finden von Dateien auf einer Kassette, ein Text-Verschlüsselungs-Programm und schließlich ein Programm, mit dem man ein „Wettrechnen" gegen den Computer veranstalten kann. Ganz offensichtlich ist der Anteil der Spiele also recht hoch. Bei einem Leseversuch mit dem PET der FUNKSCHAU-Redaktion ergaben sich zunächst Lesefehler, die erst durch Bohren eines Loches in das Kassettenrecorder-Oberteil und Einstellen der darunterliegenden Tonkopf-Taumelschraube beseitigt werden konnten. Die Programme benötigen normalerweise nur Computer mit 8 KByte RAM. Man darf sicher gespannt sein, ob die Mikrocomputer-Anwender bereit sind, den gegenüber Fachzeitschriften rund zehnmal höheren Preis pro Programm zu bezahlen, um sich ein wenig Tipparbeit zu ersparen.

```
130 Z=100:REM ANZAHL DER DATEN
140 DIMA(Z),B(Z)
150 FORI=1TOZ:A(I)=INT(RND(1)*Z)+1:NEXT
160 PRINT:PRINTCHR$(147)"UNSORTIERT :":PRINT
170 FORI=1TOZ:PRINTA(I),:NEXT
180 PRINT:PRINT
190 REM SORTIERVORGANG
200 PRINT"SORTIERBEGINN ";:T=TI:REM FUER ANFANGSZEIT
210 FORI=1TOZ:B(A(I))=A(I):NEXT
220 T=TI-T:REM FUER ENDZEIT
230 PRINTCHR$(18)"SORTIERVORGANG BEENDET"
240 PRINT:PRINT"ZEITVERBRAUCH :"INT(T/60*100+.5)/100"SEKUNDEN"
250 PRINT:PRINT"SORTIERT :":PRINT
260 FORI=1TOZ:PRINTB(I),:NEXT
270 PRINT:PRINT
280 REM AUSDRUCK OHNE NULLEN
290 PRINT:PRINT"DASSELBE OHNE NULLEN :":PRINT
300 FORI=1TOZ:IFB(I)=0THEN320
310 PRINTB(I),
320 NEXT
330 PRINT:PRINT
340 REM INDIZES ZUSAMMENFASSUNG
350 PRINT"  ALLE NICHT BELEGUNGEN STREICHEN"
360 FORI=1TOZ:IFB(I)=0THEN380
370 S=S+1:A(S)=B(I)
380 NEXT
390 REM AUSDRUCK DER REGENERIERTEN DATEI
400 PRINT:PRINT
410 PRINT"REGENERIERTE DATEI:":PRINT
420 FORI=1TOS:PRINTA(I),:NEXT
430 PRINT:PRINT
440 PRINT"ODER ANDERS GESCHRIEBEN :":PRINT
450 FORI=1TOS:PRINT"A("I")="A(I),:NEXT
460 PRINT:PRINT:PRINT"INSGESAMT"S"VERSCHIEDENE WERTE"
```

Der Trick bei dieser Sortier-Routine: Die zu sortierenden Zahlen eines Feldes dienen selbst als Index für ein zweites Feld. Das eigentliche Sortieren findet in Zeile 210 statt; der Rest des Programms dient zur Demonstration des Ablaufs mit Zufallszahlen

„Compreter" und „Interpiler" auf dem Vormarsch

Selbst in der allerjüngsten Fachliteratur wird bei der Betrachtung jener Teile von Computer-Betriebssystemen, die Hochsprachen-Programmtexte in Maschinensprache-Anweisungen umsetzen, noch immer streng zwischen „Compilern" und „Interpretern" unterschieden. Der Autor des folgenden Beitrags hält eine solche Betrachungsweise für überholt und verweist darauf, daß viele moderne Interpreter kompilieren und daß eine ganze Anzahl neuzeitlicher Compiler-Systeme einen versteckten Interpreter enthält.

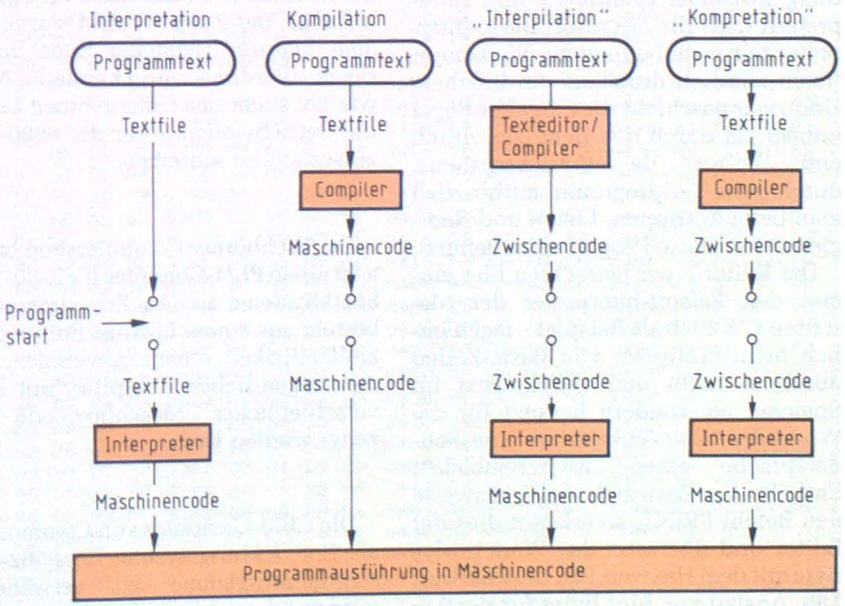
Alle Computerprogramme mit Ausnahme von solchen, die unmittelbar in Maschinencode formuliert sind, haben eines gemeinsam: Sie sind von Haus aus unverdauliche Kost für die CPU des Rechners und bedürfen erst einmal der Aufbereitung, ehe der Computer tun kann, was das Programm von ihm verlangt. Dieses Aufbereiten nun erledigt das Betriebssystem des Rechnerns selbsttätig, und wie das geschieht, dafür haben sich in der Vergangenheit zwei ursprünglich grundverschiedene Verfahrensweisen entwickelt, das „Kompilieren" und das „Interpretieren", ausgeführt durch computerinterne Dienstprogramme, den „Compiler" und den „Interpreter". Beide Bezeichnungen sind von englischen Tätigkeitswörtern abgeleitet; „to compile" heißt „zusammensetzen", während „to interprete" hier am besten mit „auslegen" zu „interpretieren" ist. Und in der Tat: Beide Bezeichnungen geben in anschaulicher Weise wieder, wodurch sich beide Übersetzungsverfahren Hochsprache nach Maschinensprache unterscheiden.

Der Compiler nimmt sich, Stück für Stück, den Text des Hochsprachen-Programms vor, übersetzt die Anweisungen in die Maschinensprache der CPU und legt das Ergebnis seiner Bemühungen dann ab: Man erhält ein Maschinen-Sprache-Programm, das man nach Abschluß der Compilierung starten kann. Auch der Interpreter nimmt sich, Portion für Portion, den Text des Hochsprachen-Programms vor und übersetzt ihn. Allerdings wird hier jede übersetzte Anweisung sofort ausgeführt, keine Aufzeichnung des Maschinencodes bleibt aufbewahrt.

Auf den ersten Blick scheint es, als ob die Unterschiede zwischen beiden Verfahren für den Benutzer ohne große Bedeutung wären: Was schert's mich, wie der Computer meine Programme umsetzt, wenn er's nur fehlerfrei tut: Sein Bier, nicht meines – möchte man sagen. Bei näherem Hinsehen allerdings zeigen sowohl Compiler als auch Interpreter Stärken und Schwächen, die sich durchaus der Außenwelt mitteilen.

Zum Ablegen des kompilierten Maschinencodes muß der Rechner nämlich auf einen dritten, oft umfangreichen Speicherbereich zurückgreifen können, den das System neben den Speicherräumen für den Hochsprachen-Text und für den Compiler selbst zusätzlich zur Verfügung zu stellen hat. Auf größeren Maschinen stört das nicht, wohl aber auf Mikrocomputern mit ihrem in aller Regel karg bemessenen Speichervorrat. Diesem Nachteil steht als typischer Compiler-Vorzug die Tatsache gegenüber, daß der kompilierte Code, im Endeffekt ja nichts anderes als ein Maschinensprache-Programm, extrem schnell abläuft – weil sich der Rechner bei der Job-Bearbeitung um keinerlei Nebenarbeiten zu kümmern hat, wie dies leider beim Interpretieren erfolgen muß.

Dabei nämlich ist der Rechner die meiste Zeit damit beschäftigt, den Hochsprachen-Text zu analysieren, Tabellen („Wörterbücher" Hochsprache-Maschinensprache sozusagen) abzusuchen und Syntax-Kontrollen durchzuführen. Für den Job selbst bleibt gemeinhin nur ein Bruchteil der Gesamt-Programm-Laufzeit übrig, der zwischen weniger als einem bis zu etwa 10 Prozent beträgt. Dem Nachteil eines betulichen Arbeitstempos steht hier jedoch als ganz beachtlicher Vorteil ge-



Interpreter und Compiler sind verwandt: Die Ablaufpläne zeigen die Ähnlichkeit. Die Erstellung des Textfiles geschieht in jedem Falle mit einem Editor, der beim „Interpiler" gleich mit der Zwischencode-Erzeugung durch den Compiler kombiniert ist

genüber, daß man weniger Schreib-Lese-Speicher braucht, oder, anders ausgedrückt, daß man bei konstanten Speicherraum sehr viel größere Programme bearbeiten kann, als würde kompiliert.

Das ist auch der Grund dafür, warum „Anfänger-Maschinen" wie TRS-80 und PET über fest eingebaute Interpreter verfügen (etwas anspruchsvollere Rechner, wie die Maschinen der Apple-Klasse, lassen dem Benutzer die Wahl, ob er residente Interpreter einsetzen oder aber nichtresidente Interpreter oder gar Compiler laden will; allerdings sind die Grenzen hier fließend. So bietet neuerdings ein amerikanisches Software-Haus auch einen Basic-Compiler für den TRS-80 an).

Stichwort fließende Grenzen: Wünschenswert wäre ein Betriebssystem, das alle Vorzüge von Compilern und Interpretern in ein und demselben Übersetzungs-Dienstprogramm gleichzeitig vereinigt. Leider muß dieser Wunsch schon aus prinzipiellen Gründen in dieser totalen Form für alle Zeiten unerfüllbar bleiben. Was jedoch realisierbar ist: Ein Kompromiß, der wengstens einen Teil der Vorzüge beider Verfahren unter einen Hut bringt. Diesen Weg nun ging beispielsweise das amerikanische Software-Unternehmen Microsoft, das die Basic-„Interpreter" für Dialekte wie Commodore-Basic, Level II, Applesoft, Palsoft und Siemens PC 100-Basic der Rechner PET, TRS-80, Apple, ITT 2020 und AIM/PC 100 schrieb.

Das Wort Interpreter ist oben aus gutem Grund in Anführungszeichen gesetzt, denn die klassische Unterscheidung zwischen Compilern und Interpretern trifft für Microsoft-Basic-Interpreter nur noch bedingt zu. Sie kompilieren nämlich durchaus ein bißchen. Und zwar geschieht dies, in aller Regel unbemerkt durch den Benutzer, durch den „Editor" des Betriebssystems, durch jenes Teilprogramm mithin, daß man beim Eintippen, Listen und Redigieren von Basic-Programmen benutzt.

Der Editor – wir betrachten hier einmal den Palsoft-Interpreter der Maschine ITT 2020 als Beispiel – legt nämlich beim Eintippen von Basic-Zeilen durchaus nicht den vollen Text im Speicher ab, sondern benutzt für die Worte des Basic-Vokabulars eine Sondersprache, einen „Zwischencode". Enthält eine Basiczelle beispielsweise den Befehl PRINT, so erkennt dies der Editor und übersetzt das Wort in das Byte mit dem Hexwert BA, Dezimalwert 186. Anstatt von fünf Bytes für die ASCII-Zeichen PRINT wird in Zwischencode also nur ein einziges Byte, hex BA, abgespeichert. Warum merkt man davon nichts? Weil beim Listen der Editor hex BA brav in PRINT zurückübersetzt. Es bleibt festzuhalten: Microsoft-Maschinen compilieren mit Hilfe ihres Editors den Original-Text in Zwischencode, das spart Speicherplatz. Danach interpretiert der eigentliche Interpreter nur noch den komprimierten Zwischencode: Das spart Rechenzeit – zwei Fliegen mit einer Klappe geschlagen, und: die Grenzen zwischen dem Compilerverfahren und dem Interpreter-Verfahren verschwimmen!

Seit neuestem gibt es im Mikrocomputer-Bereich neben den Interpreter-Compilern und Compiler-Interpretern, zum Beispiel das Betriebssystem UCSD PASCAL II. 1 der Universität von San Diego für Apple und 2020. Der Hochsprachen-Programmtext wird, bevor man ihn abfahren kann, erst einmal im klassichen Sinne kompiliert. Das Ergebnis dieser Operation freilich, und dies widerspricht nun aller Compiler-Praxis, ist durchaus kein Maschinencode-Programm, sondern ein Programm, formuliert wieder in einem Zwischencode. Spätestens hier wird deutlich, daß die in der Theorie der Computer-Wissenschaften noch immer so hochgehaltene Unterscheidung zwischen den Übersetzungsverfahren Kompilieren und Interpretieren allmählich sinnlos wird. Um den durch Compiling gewonnenen Zwischencode nämlich überhaupt weiterverarbeiten zu können, bedürfen derartige „Compiler"-Systeme hinwiederum eines eingebauten Interpreters!

Halten wir ein weiteres Mal fest: Neben Compilern und Interpretern, den Stammvätern der Hochsprachen-Übersetzern also, machen sich nun Nachkommen breit, für die es noch keine Namen gibt, aber die man wohl „Interpiler" und „Kompreter" nennen könnte. Nutznießer dieser Entwicklung sind die Microcomputer-Benutzer, deren Maschinen mit den geschilderten Mischformen zu einer Leistungsfähigkeit gelangten, die mit reinrassigen Übersetzungsverfahren nie zu erreichen wäre. Freilich: So mancher in Ehren ergraute Informatik-Professor, der seine Erstsemester in Zwischenprüfungen mit heimtückischen Fragen nach den Unterschieden zwischen Kompilation und Interpretation zu löchern pflegt, der wird nun umlernen müssen. „Interpiling" und „Compreting" für Mikrocomputer, die große Stärke von TRS-80, PET, Apple und 2020, ist heute allerdings noch nicht selbstverständlich. Da gibt es nämlich durchaus noch sehr noble und sehr teure Tischrechner auf dem Markt, die Basic-Texte in externen Speichermedien nach alter Väter Sitte ablegen: ASCII-Zeichen um ASCII-Zeichen...

Hans-Georg Joepgen

µC-Nachrichten

Für den Mikrocomputer **Eurocom-2** ist in Kürze auch der Basic-Interpreter lieferbar. Das Pascal-Projekt wurde von dem Mainzer Hersteller Eltec inzwischen allerdings zurückgestellt. Nach wie vor sucht das Unternehmen Leute, die Betriebssoftware für die 6800-Prozessorfamilie schreiben.

□

Der Eschborner Computershop bietet jetzt einen **PL/1-Compiler** für Z-80- und 8080-Systeme an. Das Programmpaket besteht aus einem Linkage Editor, dem Laufzeitpaket, einem Assembler und dem eigentlichen Compiler, mit dem verschieblicher Maschinencode erzeugt werden kann.

□

Die CBM-Computer von Commodore als Entwicklungssystem für 6502-Maschinenprogramme zu verwenden, wird mit dem „Monitor Utility System" **MOUSY** von Technofor (Lechnerstr. 8, 8026 Ebenhausen) erleichtert. Dabei handelt es sich um ein Steckmodul mit „Teachware" auf Diskette.

□

Apple-II-Besitzer können nun auch in **FORTRAN** programmieren. Dabei handelt es sich um den ANSI-Subset 77. Editor, Linker, Filer und Assembler können mit dem FORTRAN-Compiler benutzt werden, der wie bei Pascal im p-Code (Pseudo-Code) arbeitet. Das Programmpaket gibt's direkt von Apple Co.

□

Von bfi elektronik (Postfach 2098, 6057 Dietzenbach 2) können spezielle **Kompaktkassetten** für Mikrocomputer-Anwendungen (Hersteller: Verbatim) bezogen werden, die kein Vorlaufband und 2 x 5 min Laufzeit haben. Der Löschschutz an der Rückseite kann von Hand „entsichert" werden. Trotzdem sollte man, wie Versuche der FUNKSCHAU-Redaktion zeigen, den Bandanfang möglichst nicht benutzen, weil hier große Verschmutzungs-Gefahr besteht.

Herbert Müller, DB1UU

SSTV mit dem AIM-65

SSTV bedeutet „Slow Scan Television" und ist ein in Funkamateurkreisen übliches Verfahren zur Übertragung von Standbildern mit einer Bandbreite von weniger als 3 kHz. Ein Bild dauert rund acht Sekunden. Das Programm, das wir hier für die Mikrocomputer AIM-65 und PC-100 abdrucken, ist wohl eines der ungewöhnlichsten überhaupt: Es gestattet nicht nur das Senden von Texten als SSTV-Bild, sondern auch das Empfangen beliebiger Bilder mit ihren Grauwerten auf dem Thermodrucker.

SSTV verwendet ein Raster von 120 x 120 Bildpunkten, die in 7,2 s als Ganzbild (ohne Halbbild-Zeilenspur) übertragen werden. Dabei wird ein frequenzmodulierter, niederfrequenter Hilfsträger verwendet. Seine Frequenz entspricht dem jeweiligen Grauwert.

Das hier vorgestellte Programm (*Bild 1*) erfordert die 4-KByte-Version des AIM-65 und besteht grob aus zwei Teilen: einem Sendeprogramm („SSTV-Keyboard"), das maximal 20 einzutippende Buchstaben als fertiges Nf-Signal nach der SSTV-Norm generiert und somit direkt einen Kurzwellen- oder UKW-Sender modulieren kann, sowie einem Empfangsteil, der wiederum empfangene Nf-Signale direkt verarbeiten kann und auf dem Thermodrucker des AIM-65 als grauwert-gestaffeltes SSTV-Bild ausgibt.

Das Sendeprogramm

Das vom Sendeprogramm erzeugte Bild enthält vier Textzeilen mit je fünf Buchstaben (weiß auf schwarzem Grund). Die Ausgabe erfolgt innerhalb einer Interrupt-Routine, während das eigentliche Hauptprogramm nach der Initialisierung den AIM-Texteditor zur Eingabe der Zeichen aufruft.

Der AIM-Zeichengenerator

Zur grafischen Darstellung eines Zeichens verwendet das Programm den im Monitor-ROM (ab Adresse F2E1) „eingebauten Charaktergenerator", mit dem ursprünglich die 5x7-Matrix des AIM-Druckers erzeugt wird. Der entsprechende ROM-Bereich ist in fünf Blöcke für die fünf Matrix-Spalten unterteilt; jeder Block besteht aus 64 Bytes, sortiert nach den 64 darstellbaren Zeichen. Die einzelnen Bytes enthalten die Information über die sieben Matrixpunkte der jeweiligen Spalte, wobei das LSB den obersten Punkt der Spalte repräsentiert und das MSB unbenutzt bleibt. Die er-

```
0000 78 A9 29 85 020C CF A5 D0 ED 029C 00 85 D7 85 033C A9 40 8D 0B 03CC D3 A2 13 20 045C 48 A5 DD C9
0004 D8 A9 78 85 0210 07 A0 85 D0 02A0 D6 E6 D6 A9 0340 A0 8D 0B A8 03D0 7B EF 18 90 0460 90 D0 0D AD
0008 D5 A9 00 85 0214 90 02 68 40 02A4 FB 85 DC C6 0344 A2 20 A0 04 03D4 94 85 D2 A9 0464 0D A8 29 10
000C CF 85 D0 85 0218 A5 D1 8D 06 02A8 D6 D0 11 A9 0348 8E 04 A4 8C 03D8 02 85 D3 A9 0468 F0 57 A9 FF
0010 D3 8D 04 A4 021C A0 A5 D2 8D 02AC 03 85 D6 06 034C 05 A4 A2 70 03DC B0 E8 F0 ED 046C A0 08 D0 08
0014 A9 02 8D 05 0220 07 A0 A5 D3 02B0 D7 D0 09 E6 0350 A0 02 8E 04 03E0 A0 FF A6 D9 0470 C9 FE D0 1E
0018 A4 85 D4 A9 0224 85 CF A5 D4 02B4 D7 A5 DC 18 0354 A8 8C 05 A8 03E4 D0 03 A6 DA 0474 A9 00 A0 0C
001C C0 8D 0B A0 0228 85 D0 58 8A 02B8 69 05 85 DC 0358 A2 F0 A0 01 03E8 C8 C8 A5 D2 0478 85 D4 84 D5
0020 8D 0E A0 A9 022C 48 98 48 A6 02BC E6 D9 D0 04 035C 8E 04 A0 8C 03EC 18 69 13 85 047C A9 08 85 D6
0024 9F 85 D1 8D 0230 D5 A4 D8 C0 02C0 C6 DB D0 1B 0360 05 A0 A9 C2 03F0 D2 B0 0B 8A 0480 A4 DE C8 C0
0028 04 A0 A9 01 0234 29 D0 1B A9 02C4 A5 D9 C9 05 0364 8D 0E A0 EA 03F4 11 DB 91 DB 0484 0E 90 02 A0
002C 85 D2 8D 05 0238 00 E0 78 D0 02C8 D0 08 A9 FF 0368 EA A9 FB 8D 03F8 C0 0E 90 EC 0488 05 C4 DC F0
0030 A0 58 60 20 023C 02 85 D5 85 02CC 85 D9 E6 DC 036C 80 A4 AD 82 03FC EA EA A5 DB 048C 02 84 DE 18
0034 00 00 4C CF 0240 D8 E6 D5 A9 02D0 D0 1A A5 DA 0370 A4 C9 FD F0 0400 18 69 10 85 0490 90 20 A0 00
0038 F6 20 00 00 0244 9F 85 D1 A9 02D4 18 69 40 85 0374 08 C9 7F D0 0404 DB C9 90 90 0494 B1 DD 8D 00
003C A5 E3 D8 18 0248 01 85 D2 A2 02D8 DA A5 DB 69 0378 0F A2 E1 D0 0408 10 A9 FF 85 0498 A8 C8 B1 DD
0040 69 15 85 DF 024C 72 A0 13 4C 02DC 00 85 DB A4 037C 08 A2 90 86 040C DB A5 DA C9 049C 8D 01 A8 A9
0044 85 E1 A5 E4 0250 04 03 E0 01 02E0 DC B1 DD 29 0380 D5 86 DD A2 0410 01 F0 06 46 04A0 00 91 DD 88
0048 69 00 85 E0 0254 D0 1B C0 11 02E4 3F A8 A5 D7 0384 C1 8E 0C A8 0414 D9 66 DA E6 04A4 91 DD C6 D6
004C 85 E2 4C D5 0258 B0 1D A5 E0 02E8 31 DA D0 0C 0388 A6 DB A9 FF 0418 DB 4C 69 03 04A8 D0 08 A9 08
0050 F7 A9 40 2D 025C 85 DE A5 DF 02EC A2 54 A0 01 038C 85 D2 A9 40 041C EA EA EA EA 04AC 85 D6 A9 04
0054 0E A0 F0 0D 0260 85 DD C5 E1 02F0 D0 0A A2 9F 0390 2C 0D A8 F0 0420 48 A9 02 2C 04B0 85 D5 A5 D4
0058 A6 D5 CA D0 0264 D0 08 E9 15 02F4 A0 01 D0 04 0394 FB AD 04 A8 0424 0D A0 F0 24 04B4 D0 07 E6 DD
0268 85 DD B0 02 02F8 A2 E2 A0 00 0398 A5 D8 D0 3F 0428 AD 08 A0 85 04B8 E6 DD 18 90
005C FB 8D 0E A0 026C C6 DE 4C F2 02FC 86 D1 84 D2 039C A5 D7 C9 50 042C D7 AD 09 A0 04BC 04 C6 DD C6
0060 A9 00 8D 0B 0270 02 E0 0D 90 0300 A2 4A A0 05 03A0 B0 33 A4 D3 0430 85 D8 A9 B8 04C0 DD 68 A8 68
0064 A0 20 F1 E6 0274 77 E0 6D B0 0304 86 D3 84 D4 03A4 F0 07 C6 D3 0434 8D 08 A0 A9 04C4 40
0068 4C 20 03 0278 73 C0 23 B0 0308 E6 D8 68 A8 03A8 E8 D0 30 F0 0438 01 8D 09 A0 0D90 2D 46 55 4E
00DF 90 0D A5 0D 027C 6F C0 06 90 030C 68 AA 68 40 03AC BC A9 C6 A0 043C AD 01 A0 AD 0D94 4B 53 43 48
00E3 90 0D 00 10 0280 6B D0 39 A9 0320 A9 05 85 D3 03B0 02 84 D9 A0 0440 0C A0 49 01 0D98 41 55 2D 53
010A 3E 0A 4C 33 0284 FF 85 D9 A5 0324 85 D5 85 DC 03B4 00 84 DB A6 0444 8D 0C A0 68 0D9C 53 54 56 2D
010E 00 4C 39 00 0288 DC 38 E9 04 0328 85 DE A9 FF 03B8 DA 84 DA F0 0448 40 EA EA EA 0DA0 50 52 47 4D
0112 4C 51 00 028C 85 DC A9 E1 032C 85 DB A9 90 03BC AC CA D0 0D 044C AD 04 A0 58 0DA4 0D 00
0200 48 D8 2C 04 0290 85 DA A9 F3 0330 85 DD A9 E1 03C0 A4 DC C8 C0 0450 A5 D5 F0 07
0204 A0 A5 CF 38 0294 85 DB E0 0D 0334 8D 0C A8 A9 03C4 0E 90 02 A0 0454 20 BD F0 C6
0208 ED 06 A0 85 0298 D0 0D EA A9 0338 00 8D 0C A0 03C8 05 84 DC A9 0458 D5 10 68 98
```

Bild 1. Programmlisting für Senden und Empfangen von Slow-Scan-Television-Bildern mit AIM-65 (Rockwell) oder PC-100 (Siemens)

sten sechs Bit des darzustellenden ASCII-Zeichens dienen als Index, um das zugehörige Byte innerhalb eines Blockes zu adressieren. Durch eine UND-Verknüpfung (02E8) selektiert man das für die jeweilige Reihe maßgebliche Bit heraus. Für die erste Reihe ist also die dazu erforderliche Maske 01_{hex}. Zur nächsten Reihe gelangt man jeweils durch einen Links-Shift dieser Maske (02AF), zu dem die nächste Spalte definierenden Byte durch Addition von 64 (40_{hex}) zur Basisadresse der UND-Verknüpfung (02D2–02DD).

Das „Timing"

Die Schrift ist bei der gewählten Formatierung von vier Textzeilen zu je fünf Buchstaben auch noch bei schlechterem Empfangssignal lesbar. Rückt man noch etwas vom Bildrand ein, kommt gerade auf drei Bildpunkte (und -Zeilen) ein Matrixpunkt, der dann 1,36 ms lang ausgegeben werden muß. Je nach Grauwert entspricht das nur etwa zwei bis drei Periodenlängen der zugehörigen Frequenz, weshalb sich Phasensprünge oder Verzögerungen beim Frequenzwechsel bereits störend auswirken. Im vorliegenden Programm sind daher die Frequenzen für schwarz und weiß gerade so gewählt, daß ein ganzzahliges Vielfaches der Periodendauer genau der jeweiligen Ausgabedauer entspricht (1462 Hz und 2193 Hz). Das gleiche gilt für die Synchronimpulse von 1200 Hz (6 Perioden in 5 ms). Die Zeit wird durch Zählung der Perioden gemessen, so daß man für die SSTV-Signal-Erzeugung mit einem einzigen Timer auskommt.

Das Programm

Der frei laufende Timer 1 löst nach jeder halben Periode einen Interrupt aus. Zu Beginn der Interrupt-Routine (ab 0200) wird die erforderliche Tonlänge minus 2 (CF, D0) um den Inhalt der Timer-Latches verringert. Ist das Ergebnis negativ, dann ist die Zeit beim nächsten Interrupt abgelaufen. In diesem Fall lädt das Programm zunächst die bereits in einem vorhergehenden Durchgang der Interrupt-Routine in der Zeropage abgelegten Werte für die neue Frequenz (D1, D2) und Tondauer (D3, D4) in die Latches bzw. in den Zeitzähler. Bevor nun die nächsten Werte bestimmt werden, wird das Interrupt-Disable-Flag gelöscht, um erneuten Interrupt zur Periodenzählung zuzulassen. Das richtige Bildformat ergibt sich durch Zählen der Zeilen (D5) und der Punkte innerhalb einer Zeile (D8). Mit Hilfe des Zählers auf drei (D6) wird die Bitmaske für die Matrixreihe (D7) nur jede dritte Zeile nach links geschoben, der Zähler auf sechs (D9) sorgt dafür, daß innerhalb einer Zeile nach fünf Matrixpunkten und einem Punkt Abstand ein neuer Buchstabe ausgegeben wird, d. h. der Index (DC) zum Text-Pointer (DD, DE) um eins erhöht wird. Mit diesem Vektor holt sich das Programm den auszusendenden Text aus dem Texteditor.

Die Bedienung

Vor dem Start des Programmes muß zunächst eine Standardtext-Zeile eingeschrieben werden. Hierzu initialisiert man den Texteditor, wobei im Hinblick auf das Empfangsprogramm eine Textanfangsadresse ab 0D90 möglich ist, und schreibt eine Standardzeile von genau 20 Buchstaben Länge ein, die jeweils am Anfang ausgesendet werden soll, z. B. Rufzeichen und Standort usw. Wichtig ist, daß grundsätzlich 20 Zeichen pro Zeile eingegeben werden, also u. U. mit Blanks auffüllen, bis der Cursor gerade verschwindet. Die längeren weißen Teilstriche auf der Display-Abdeckung des AIM entsprechen hierbei gerade der Aufteilung in vier Textzeilen im SSTV-Bild. Anschließend wird die Eingabe durch zweimal ‚Return' beendet und der Editor über Q oder Escape verlassen. Am besten speichert man diese Standardzeile gemeinsam mit dem Programm (also in Maschinensprache) auf Kassette ab, indem man zusätzlich den Bereich DF-E6 (Editorvariablen) und die vom Text belegten Bytes angibt oder die Dump-Routine aus Bild 2 verwendet.

Nachdem man sich vergewissert hat, daß der AIM-Drucker abgeschaltet ist, kann das Programm nun mit den Tasten F1 oder F2 gestartet werden. In beiden Fällen wird zunächst der IR-Betrieb durch T1 ermöglicht und, beginnend mit einem Bildsynchronimpuls, der Inhalt der Standardzeile gesendet. Erfolgte der Start mit F2, befindet man sich nun im Readmodus des Editors, und es können so hinter die bereits vorhandene erste Zeile weitere Textzeilen mit je 20 Zeichen Länge eingeschrieben werden. Im Bild erscheint jeweils der Inhalt der letzten Zeile, die zum Zeitpunkt des Bildsynchronimpulses mit ‚Return' abgeschlossen ist. Am Ende der Eingabe wird der Editor wieder durch zweimal Return und Q oder Escape verlassen. Danach kann man die Bildaussendung mit F3 beenden, worauf das Programm das Bildende abwartet und erst dann Timer 1 anhält. Auf dem Display erscheint OFF, und es erfolgt ein Rücksprung ins Monitor-Programm. Um ein Überlaufen des Texteditors zu vermeiden, bleibt bei jedem Neueinschreiben mit ‚F2' nur die erste Zeile erhalten.

Der eingegebene Text kann jedoch mit F1 nochmals abgerufen werden. Die Aussendung beginnt dann wieder mit der Standardzeile, die auch auf dem Display erscheint. Nun können die Editorfunktionen D, B, U, T und F verwendet werden. Die Übernahme der angezeigten Zeile in das SSTV-Bild erfolgt während des Bildsynchronimpulses. Durch Q oder ‚Escape' kann man den Texteditor verlassen und dann mit F3 die Ausgabe beenden.

Das Empfangsprogramm

Zwei Umstände erlauben eine wohl erstaunliche Zweckentfremdung des AIM-Druckers: zum einen die softwaremäßige Steuerung des Druckvorganges, auf der anderen Seite die Tatsache, daß der Ausdruck einer Zeile etwa die gleiche Zeit benötigt wie die Aussendung. Außerdem ergeben zufälligerweise 120 Zeilen gerade ein annähernd quadratisches Feld auf dem Drucker. Die Möglichkeit des gleichzeitigen Ausdruckens reduziert die notwendige Zwischenspeicherung auf ein Minimum, so daß die 4 KByte des AIM (die 1-K-Variante kann ja leicht erweitert werden) für Sende- und Empfangsprogramm mehr als ausreichen. Da die zehn Thermo-Druckköpfe per Programm nur ein- oder ausgeschaltet werden können, muß zur Darstellung verschiedener Helligkeiten die Druckdauer für die einzelnen Punkte variiert werden. Hierfür sind acht verschiedene Zeiten wählbar, was einer Wiedergabe von max. neun Graustufen entspricht. Die Auflösung ist durch die Länge der einzelnen Programmteile auf 90 Punkte pro Zeile begrenzt, d. h. neun Punkte für jeden Druckknopf. Die sich daraus ergebende Bildqualität ist immerhin gut genug, um nicht nur Schrift, sondern z. B. auch das Konterfei des QSO-Partners noch deutlich erkennen zu können.

Drucker-Belastung

Die Belastung der Thermo-Köpfe ist natürlich speziell bei dunklen Bildteilen größer als bei der normalen Verwendung. Zur Schonung werden diese daher nach jedem Punkt für knapp 2 ms abgeschaltet. Außerdem arbeitet man, wie weiter unten noch näher beschrieben wird, mit einer verringerten Drukkerspannung. Wer dennoch seine Bedenken hat, den beruhigt vielleicht die Tatsache, daß das Testmodell den Ausdruck von insgesamt mehreren Rollen

Druckerpapier unbeschadet überstanden hat. Freilich kann keine Garantie gegeben werden, und die Bedienungshinweise, die noch folgen, sollte man auf alle Fälle beachten. Falls ein Printer-Element ständig eingeschaltet bleibt oder ein sonstiger Fehler auftritt, ist sofort mit Reset zu stoppen.

Funktionsweise

Zunächst kurz zur Funktionsweise des Empfangsprogrammes ab 0320: Um ohne externe Bauteile auszukommen, wird das SSTV-Nf-Signal direkt am Kassettenrecordereingang des AIM eingespeist (AUDIO IN, J1, L). Damit die Decodierung im Interrupt-Betrieb erfolgen kann, ist eine zusätzliche Verbindung zwischen dem Anschluß X der Steckerleiste J3 (Expansion) und Anschluß 20 von Steckerleiste J1 (Application) herzustellen. Das rechteckförmige Eingangssignal gelangt hierdurch an den Eingang CA 1 des User-VIA. Mit der Interruptroutine ab 0428 kann nun mit Hilfe von Timer 2 die (halbe) Periodendauer der Niederfrequenz gemessen werden: Eine Flanke an CA1 löst den Interrupt aus, der Meßwert wird in die Zero-Page übertragen und Timer 2 neu gestartet.

Das Hauptprogramm berechnet aus dem jeweils letzten Meßwert den entsprechenden Grauwert und speichert ihn bereits so ab, wie er für die Ausgabe an den Drucker benötigt wird. Hierbei ist zu beachten, daß die Zeile seriell empfangen, aber in zehn Sektoren gleichzeitig ausgedruckt wird. Außerdem selektiert das Hauptprogramm die Synchronimpulse heraus und zeigt auf dem Display an, ob sie im richtigen zeitlichen Abstand erkannt werden.

Die Ansteuerung des Druckers geschieht wiederum per Interrupt, in regelmäßigen Abständen durch den frei laufenden Timer 1 ausgelöst. In dem Programmteil ab 044C werden die einzelnen Punkte an die Thermoköpfe ausgegeben und im Speicher wieder gelöscht. Da abwechselnd von links nach rechts und von rechts nach links gedruckt wird, ist jeweils am Zeilenende die Reihenfolge der Ausgabe umzukehren. Wenn die Druckschiene den rechten Anschlag erreicht – also nach jeder zweiten Zeile – gibt der Drucker einen Impuls ab, der vor der Ausgabe der nächsten beiden Zeilen abgewartet wird.

Bedienung

Nachdem man das Programm eingegeben und noch einmal mit dem Listing verglichen hat, muß nun zunächst die Druckergeschwindigkeit eingestellt werden. Da der Drucker bei einer Betriebsspannung von 24 V zu schnell läuft, ist diese auf etwa 20 V zu reduzieren[1]). Eine Feineinstellung kann dann über den Regler VR3 auf der AIM-Platine vorgenommen werden: Hierzu ändert man vorübergehend Speicherstelle 04A0 von 00 in FF um und startet das Programm bei 0320. Der Drucker ist nun, wie auch später beim Empfang, durch die Taste Print ein- und durch Escape auszuschalten. Zuerst stellt man VR3 an den linken Anschlag. Auf PRINT sollten abwechselnd zwei dunkle und zwei Leerzeilen ausgedruckt werden. Während der Drucker läuft, dreht man VR3 solange im Uhrzeigersinn, bis keine Leerzeilen mehr auftreten und auf dem Papier eine dunkle Fläche erscheint, die von neun möglichst schmalen Linien unterbrochen ist. Falls zwar keine Leerzeilen entstehen, aber jede zweite Zeile zu kurz gerät, läuft der Drucker bereits zu langsam, und VR3 ist nach links zu drehen bzw. die Spannung wieder etwas zu erhöhen. Gegebenenfalls sollte man hin und wieder mit Escape eine Pause einlegen, um den Drucker nicht zu stark zu belasten. Ist die Einstellung beendet, verläßt man das Programm über einen Reset und schreibt in 04A0 wieder 00 ein. Bei einem erneuten Start darf der Drucker jetzt nur unbedrucktes Papier ausgeben. Auch in der geänderten Einstellung kann er im übrigen für normale Textausdrucke weiter verwendet werden.

Anschließend kann, sofern man die oben beschriebene Verbindung hergestellt hat, ein SSTV-Signal (Nf) zum Kassettenrecorder-Eingang des AIM hereingespielt werden. Natürlich versucht man es zunächst mit einem möglichst ungestörten Signal, z. B. der Kassettenaufnahme eines mit dem Sende-Programm generierten Textbildes (bei der Wiedergabe vom Recorder sollte der Steuerausgang des AIM, der den Recorder schaltet, nicht wirksam sein, da sich dieser sonst beim Ausdrucken abschaltet). In jedem Fall muß der Pegel am Eingang des AIM mindestens so hoch sein wie beim Einlesen von Programmen. Die letzte Display-Stelle dient als Einstellhilfe: solange „0" erscheint, ist entweder der Pegel zu niedrig oder die Synchron-Impulse liegen nicht im Bereich von 1200 Hz. Wird „F" angezeigt, treten sie zu oft bzw. zu früh auf. Bei SSB-Empfang ist dann die Frequenz-Einstellung am Empfänger zu korrigieren. Bei richtiger Abstimmung signalisiert ein „S", daß man mit der Taste PRINT den Ausdruck beginnen kann. Grundsätzlich sollte man nach etwa zwei bis drei Bildern wieder mit ‚ESCAPE' unterbrechen, speziell bei dunklen Bildern. Wenn trotz stabilem „S" auf dem Display im Ausdruck weiße Querbalken auftreten, muß man an VR3 etwas nachgleichen, bis sie nur noch zwischen einzelnen Bildern erscheinen.

Einen gewissen Spielraum schafft das Programm dadurch, daß es bis zu neun Zeilen zwischenspeichert. Der entsprechende Buffer belegt die Seiten 05 bis 0D jeweils bis 0X90.

Mögliche Änderungen

Folgende Parameter können variiert werden: Der Wert in 03EE beeinflußt Helligkeit und Kontrast, eine kleinere Zahl (z. B. 10_{hex}) bewirkt einen dunkleren Ausdruck. Eine Änderung bei 0433 betrifft Helligkeit und Synchronimpulse. Liegt etwa das ankommende Signal in der Frequenz insgesamt zu tief, kann durch einen höheren Wert in 0433 (z. B. $C0_{hex}$) in gewissen Grenzen korrigiert werden.

Die Bildbreite ist durch 034F (70_{hex}) festgelegt. Zur richtigen Synchronisation muß das ausgedruckte Bild innerhalb des letzten Zehntels enden. Ist es schmaler, d. h. das empfangene Bild ‚zu schnell', wird ‚F' angezeigt und nichts ausgedruckt. Dann ist der Wert in 034F zu verringern. Falls dagegen das letzte Zehntel des Ausdruckes ganz ausgefüllt ist und auf dem Display kurz ‚0' aufflackert, muß eine größere Zahl eingeschrieben werden. Zum einwandfreien Empfang der langsameren amerikanischen SSTV-Sendungen wären allerdings zusätzlich Drucker-Geschwindigkeit und -Variable neu zu bestimmen.

Die Kombination mit dem Sendeprogramm ist denkbar einfach: ändert man dort den Sprungbefehl bei 0068 von JMP E182 in JMP 0320 um, so landet man mit Taste ‚F3' nicht mehr im Monitorprogramm, sondern in der Empfangsroutine. Bevor mit ‚F1' oder ‚F2' erneut auf Sendung gegangen werden kann, ist allerdings die Resettaste zu betätigen. Zum Hex-Listing noch eine Anmerkung: Der Vektor bei 010A gehört zu dem später beschriebenen SAVE-Programm.

Abschließend noch eine Ergänzung zur Senderoutine; führt man folgende Änderungen durch:

[1]) Da die an Anschluß 6 der Klemmleiste angelegte Spannung nur für den Drucker benötigt wird, kann man entweder die entsprechende Ausgangsspannung im Netzteil ändern oder einen geeigneten Vorwiderstand in die Zuleitung legen.

```
0A3E A9 54 20 76 | 0AA6 68 8D 19 A4
0A42 E8 A9 00 8D | 0AAA 20 38 E5 AD
0A46 06 01 8D 07 | 0AAE 1B A4 20 38
0A4A 01 AA 8E E1 | 0AB2 E5 AD 1A A4
0A4E 0A BD E2 0A | 0AB6 20 38 E5 20
0A52 C9 FF D0 06 | 0ABA 31 E5 A9 00
0A56 20 EB E4 4C | 0ABE 8D 15 A4 20
0A5A 82 E1 8D 1B | 0AC2 5D E5 CE 19
0A5E A4 E8 BD E2 | 0AC6 A4 D0 F0 AD
0A62 0A 8D 1A A4 | 0ACA 1F A4 20 3B
0A66 E8 BD E2 0A | 0ACE E5 AD 1E A4
0A6A 8D 1D A4 E8 | 0AD2 20 3B E5 20
0A6E BD E2 0A 8D | 0AD6 66 E5 4C 7D
0A72 1C A4 E8 8E | 0ADA 0A AE E1 0A
0A76 E1 0A A0 01 | 0ADE 4C 4F 0A 08
0A7A 20 CD E2 20 | 0AE2 01 0A 01 0B
0A7E F0 E9 20 07 | 0AE6 0A 3E 0B 0A
0A82 E9 20 4D EB | 0AEA 01 0C 01 14
0A86 AD 1C A4 38 | 0AEE 00 DF 00 E6
0A8A ED 1A A4 48 | 0AF2 00 00 00 6A
0A8E AD 1D A4 ED | 0AF6 02 00 04 C4
0A92 1B A4 D0 09 | 0AFA 0E 00 0E 15
0A96 68 F0 42 C9 | 0AFE FF FF FF FF
0A9A 18 90 05 B0 | 0B02 FF FF FF FF
0A9E 01 68 A9 18 | 0B06 FF FF FF FF
0AA2 48 20 BA E9 | 0B0A FF
```

Bild 2. Dieses Programm dürfte auch für Nicht-Funkamateure sehr nützlich sein. Es gestattet das schnelle Kopieren beliebiger Speicherinhalte, deren Adressen ab 0AE2 stehen. Das Abspeichern erfolgt so: (D)FROM = (Space) TO = (Space) OUT = U

```
(M)=02EC    A2 E2 A0 00
(M)=02F0    10 0A        und
(M)=02F8    A2 54 A0 01
```

so wird statt der inversen Darstellung jetzt dunkle Schrift auf weißem Grund gesendet.

Zero-Page-Belegung

Sendeprogramm

CF, D0 Zeitzähler
D1, D2 nächste Frequenz
D3, D4 nächste Tonlänge
D5 Zeilenzähler
D6 Zähler auf 3
D7 Reihen-Maske
D8 Punktezähler
D9 Zähler auf 6
DA, DB Spalten-Block-Adresse
DC Buchstabennummer
DD, DE Textpointer

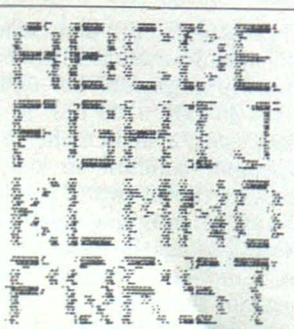


Bild 3. So sieht ein mit dem AIM-Thermodrucker empfangenes SSTV-Bild aus – es wurde mit dem AIM-Sendeprogramm erzeugt und per Funk übertragen

Empfangsprogramm

D2 Grauwert
D3 Länge Synchronimpuls
D4 Druckrichtung
D5 Zähler: Druckpause
D6 Zähler: Segmente pro Druckpunkt
D7, D8 Periodendauer
D9, DA Maske Port B, Port A (Thermoelemente)
DB, DC Pointer (Abspeichern)
DD, DE Pointer (Drucken)

Abspeichern des Programms

Für das Abspeichern des Programms mit seinen unterschiedlichen Adressenbereichen auf Kassette wurde eine Routine entwickelt, die dies weitgehend automatisch übernimmt (*Bild 2*). Sie kopiert sich selbst sowie alle Speicherbereiche, deren Adressen ab 0AEA stehen und arbeitet mit der U-Option des Dump-Befehls. (Eingabe: <D> FROM = (Space) TO = Space) OUT = U.) Selbstverständlich läßt sich das Programm in Bild 2 auch für alle anderen AIM-Programme einsetzen, wenn man die Anfangs- und Endadressen ab 0AEA einträgt und mit FF_{hex} abschließt. Bei 0AFA muß hier übrigens korrekterweise 0D90 0DA5 stehen.

Zwei Beispiele für empfangene SSTV-Bilder sind in *Bild 3* und *Bild 4* angegeben. Insbesondere Bild 4 zeigt, daß der Thermodrucker Erstaunliches zu leisten vermag!

Bild 4. Kaum zu glauben, aber dieses auf UKW übertragene SSTV-Bild wurde ebenfalls vom AIM-Thermodrucker ausgegeben!

Amateurfunk-Programme

Die folgende Übersicht soll den Funkamateuren unter unseren Lesern helfen, das passende Programm für ihren Mikrocomputer zu finden. Verwendete Abkürzungen: FS = FUNKSCHAU, PG = Sonderheft „Programme für Kleincomputer und Taschenrechner" der FUNKSCHAU, RPB 173 = „Anwendungsbeispiele für den Mikroprozessor 6502", Franzis-Verlag, HC2 = Sonderheft „Hobbycomputer 2" der FUNKSCHAU.

KIM-1

RTTY-Sende- und Empfangsprogramm	RPB 173
ASCII-Siebensegment-Sende- und Empfangsprogramm	RPB 173
RTTY-Siebensegment-Empfangsprogramm	PG
CW-Siebensegment-Decoder	PG
CW-Sendeprogramm	PG
Morsetrainer	PG
Amateurfunk-Automat (ASCII)	HC2
RTTY-Empfang mit ASCII-Interrupt-Ausgabe	HC2

AIM-65

RTTY-Empfangsprogramm	FS 15/1979
RTTY-Sendeprogramm	FS 16/1979
RTTY/CW-Konverterplatine	FS 16/1979
RTTY und CW mit Druckerausgabe	PG
CW-Sendeprogramm	FS 16/1979
CW-Empfangsprogramm	FS 9/1980
ASCII-Sende- und Empfangsprogramm	FS 18/1980
SSTV-Sende- und Empfangsprogramm	FS 14/1980
Sprachsynthese-Rufzeichengeber	FS 16/1980

Nascom-1

ASCII-Sende- und Empfangsprogramm	FS 4/1979
RTTY-Sende- und Empfangsprogramm	FS 1/1980

Basic

QTH-Kenner: Entfernungsberechnung	HC2, PG
Berechnung von Yagi-Antennen	PG
Morsedecoder in Basic	PG
Rotorsteuerung	HC2

Ethem Artikoglu

Musikprogramm für den TM 990

Das nachfolgend beschriebene Programm erlaubt einstimmige Musik mit der Mikrocomputerfamilie TMS 9900, z. B. mit dem Einplatinen-Computer TM 990/189. Dabei wird auch die typische 16-bit-Programmiertechnik deutlich.

Das vom Mikrocomputer belegte RAM gliedert sich bei unserem Musikprogramm in drei Abschnitte: Den für die 9900-Familie typischen „Workspace", in dem die Register der CPU untergebracht sind (die CPU enthält diese Register nicht selbst!); das Musik-Steuerprogramm und schließlich die codierten Musikstücke. Die notwendige Hardware besteht nur aus einem beliebigen Verstärker nebst Lautsprecher, der an den Port P O des CRU-Bausteins im Mikrocomputer zu schalten ist.

Bild 1 zeigt das vollständige Assembler-Listing des Musikprogramms. Von links nach rechts enthält er die Zeilennummer des Assembler-Editors, die hexadezimale Adressen, die hexadezimalen Operationscodes, die mnemonischen Befehlsworte nebst ihren Operanden sowie das Kommentarfeld.

Die Arbeitsregister werden dabei wie folgt verwendet:

R1: Anfangsadresse der codierten Melodie
R3: Konstante für Tonlänge (1 Takt)
R4: Konstante für Pausenlänge
R5: Daten (codierte Tonhöhe Fz)
R6: Taktzahl
R7: Zähler für Fz
R8: Zähler für Länge
R9: Zähler für Taktzahl
R10: Momentane Adresse der Melodie

Aus *Bild 2* gehen die Formeln zur Berechnung von Frequenzen und Längen hervor; *Tabelle 1* zeigt die Codierung von vier Oktaven. Damit man gleich ein kleines Demonstrationsprogramm laufen lassen kann, zeigt *Tabelle 2* die Codierung für ein Menuett.

Jedes Lied beginnt mit zwei Anfangswörtern, die das Lied charakterisieren, anschließend die „Noten-Wörter" und schließlich drei Endwörter, die eine bedingte Steuerungsfunktion auslösen *(Bild 3)*. Das erste Wort KT beeinflußt das Spieltempo zusammen mit der Taktlänge tl (vgl. Bild 2). Kp ist die zwischen den einzelnen Tönen stets vorhandene Pause, um auch bei Aufeinanderfolge gleicher Töne diese un-

```
0001                ************************************************
0002                *                                              *
0003                *                EINSTIMMIGES                  *
0004                *    MUSIKPROGRAMM FUR TEXAS INSTRUMENTS       *
0005                *                   TM990                      *
0006                *            MIKROCOMPUTERFAMILIE              *
0007                *         VON E.ARTIKOGLU (6.JUL.1979)         *
0008                *                                              *
0009                ************************************************
0010                *
0011 0000 020C      LI   R12,>0100   CRU BASE SETZEN
     0002 0100
0012 0004 C281      MOV  R1,R10      ANFANGSADR. TRANSPORTIERN
0013 0006 C0FA      MOV  *R10+,R3    INHALT DER MOMENT.ADR. NACH R3
0014 0008 13FD      JEQ  $-4         ZURUECK ZUM ANF. BEI LIEDERBLOCKENDE
0015 000A C13A      MOV  *R10+,R4    INHALT DER MOMENTANEN ADR. NACH R4
0016 000C C17A      MOV  *R10+,R5    INHALT DER MOMENTANEN ADR. NACH R5
0017 000E 13FB      JEQ  $-8         WENN 3.SCHLUSSWORT "S" IST GEHE ZUM LIED
0018 0010 C185      MOV  R5,R6       NOTENWORT TRANSPORTIEREN
0019 0012 0245      ANDI R5,>0FFF    IN R5 FREQENZZAHL "FZ" ISOLIEREN
     0014 0FFF
0020 0016 09C6      SRL  R6,12       IN R6 TAKTZAHL "TZ" ISOLIEREN
0021 0018 C243      MOV  R3,R9       ----------------------------------------
0022 001A 04C8      CLR  R8          "  DATEN VORBEREITEN UND GES.TONLAENGE
0023 001C 3E05      DIV  R5,R8       "  "TL"ERMITTELN;   TL=(KT*TZ)/FZ
0024 001E 3A06      MPY  R6,R8       ----------------------------------------
0025 0020 C204      MOV  R4,R8       PAUSENKONSTANTE "KP" TRANSPORTIEREN
0026 0022 C1C5      MOV  R5,R7       "FZ" TRANSPORTIEREN
0027 0024 1D10      SBO  16          HIGH-PEGEL SETZEN
0028 0026 0607      DEC  R7          "FZ-1"
0029 0028 16FE      JNE  $-2         "FZ=0?" NEIN SPRINGE AUF DEC R7
0030 002A C1C5      MOV  R5,R7       "FZ" TRANSPORTIEREN
0031 002C 1E10      SBZ  16          LOW-PEGEL SETZEN
0032 002E 0607      DEC  R7          "FZ-1"
0033 0030 16FE      JNE  $-2         "FZ=0?" NEIN SPRINGE AUF DEC R7
0034 0032 0609      DEC  R9          "TL-1"
0035 0034 16F8      JNE  $-18        "TL=0?" NEIN SPRINGE AUF MOV R5,R7
0036 0036 0608      DEC  R8          "KP-1"
0037 0038 16FE      JNE  $-2         "KP=0?" NEIN SPRINGE AUF DEC R8
0038 003A 10E8      JMP  $-46        SPRINGE ZURUECK U. WIEDERHOLE
0039                END
0000 ERRORS
```

Bild 1. Assembler-Listing für ein einstimmiges Musikprogramm. Es läuft mit geringen Anpassungen auf jedem System mit der 16-bit-Prozessorfamilie TMS 9900

Tabelle 1. Codierung der Töne

	f (Hz)	Note	Fz (>)
Große Oktave	65,41	C	4 7 8
	69,3	cis	4 3 8
	73,42	D	3 F B
	77,78	dis	3 C 2
	82,41	E	3 8 C
	87,31	F	3 5 9
	92,5	fis	3 2 9
	98	G	2 F B
	103,83	gis	2 D 0
	110	A	2 A 8
	116,55	ais	2 8 1
	123,47	H	2 5 D
Kleine Oktave	130,81	c	2 3 B
	138,6	cis	2 1 B
	146,8	d	1 F D
	155,57	dis	1 E 0
	164,8	e	1 C 5
	174,6	f	1 A B
	185,01	fis	1 9 3
	196	g	1 7 C
	207,66	gis	1 6 7
	220	a	1 5 3
	233,1	ais	1 4 0
	246,9	h	1 2 E
Eingestrichene Oktave	261,6	c'	1 1 C
	277,2	cis'	1 0 C
	293,6	d'	0 F D
	311,14	dis'	0 E F
	329,6	e'	0 E 1
	349,2	f'	0 D 5
	370	fis'	0 C 8
	392	g'	0 B D
	415,3	gis'	0 B 2
	440	a'	0 A 8
	466,2	ais'	0 9 F
	493,9	h'	0 9 6
Zweigestrichene Oktave	523,3	c''	0 8 D
	554,4	cis''	0 8 5
	587,3	d''	0 7 E
	622,3	dis''	0 7 6
	659,3	e''	0 7 0
	698,5	f''	0 6 9
	740	fis''	0 6 3
	784	g''	0 5 D
	830,66	gis''	0 5 8
	880	a''	0 5 3
	932,4	ais''	0 4 E
	987,8	h''	0 4 A
Pausen-Zeichen	23427,5	min.	1 0 0 1
	23437,5	max.	F 0 0 1

terscheiden zu können, und soll etwa 10...50 ms betragen. Dann folgen die Notencodierungen; sie umfassen jeweils vier Hex-Digits, wobei das höchstwertige erste die Tondauer und die drei übrigen (16 bit Wortlänge!) die Frequenz angeben. Wenn für die Frequenz 001 eingegeben wird, so liegt sie außerhalb des Hörbereichs (23 kHz) und wird somit als Pause wahrgenommen.

Konstante für Eintakt-Tonlänge

$$K_T = \frac{t_L}{1,\overline{3} \cdot 10^{-5}}$$

Eintakt-Tonlänge

$$t_L = K_T \cdot 1,\overline{3} \cdot 10^{-5}\ [\text{sec}]$$

Konstante für Pause zwischen zwei Noten

$$K_P = \frac{t_P}{6,\overline{6} \cdot 10^{-6}}$$

Pausenlänge

$$t_P = K_P \cdot 6,\overline{6} \cdot 10^{-6}\ [\text{sec}]$$

Frequenz-Zahl

$$F_Z = \frac{1}{f \cdot 1,\overline{3} \cdot 10^{-5}} - 2,2$$

Ton-Frequenz

$$f = \frac{1}{1,\overline{3} \cdot 10^{-5}\,(F_Z + 2,2)}\ [\text{Hz}]$$

Ges. Tonlängen-Zahl für T_Z Takte

$$T_L = \frac{K_T}{F_Z} \cdot T_Z$$

Bild 2. Formeln für die Berechnung der unterschiedlichen vom Programm verwendeten Konstanten

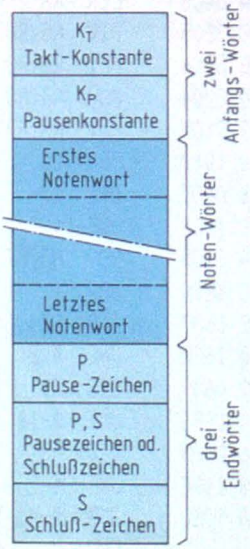


Bild 3. Jedes Lied setzt sich aus zwei Anfangswörtern, den Notencodes und drei Schlußwörtern zusammen

Die drei Schlußwörter setzen sich aus einem Pausenwort (X001), einem weiteren Pausenwort oder einem Schlußwort und schließlich einem Schlußwort (0000) zusammen. Wenn das zweite Wort ein Schlußzeichen ist, wird wieder von vorn begonnen; wenn es ein Pausenwort ist, so fährt der Computer mit dem unmittelbar folgenden Lied fort.

Tabelle 2. Codierung für ein Menuett

Note	Tz HEX	Code HEX
	K_T =	2 8 0 0
	K_P =	2 0 0 0
f''	4	4 0 6 9
c''	4	4 0 8 D
h'	4	4 0 9 6
a'	4	4 0 A 8
g'	4	4 0 B D
f'	4	4 0 D 5
g'	4	4 0 B D
f'	4	4 0 D 5
e'	4	4 0 E 1
f'	12	C 0 D 5
a'	4	4 0 A 8
a'	4	4 0 A 8
a'	4	4 0 A 8
h'	6	6 0 9 6
d''	2	2 0 7 E
h'	4	4 0 9 6
g'	4	4 0 B D
g'	4	4 0 B D
g'	4	4 0 B D
a'	6	6 0 A 8
c''	2	2 0 8 D
a'	4	4 0 A 8
h'	4	4 0 9 6
h'	4	4 0 9 6
a'	4	4 0 A 8
g'	12	C 0 B D
g'	4	4 0 B D
e'	4	4 0 E 1
c'	4	4 1 1 C
a'	4	4 0 A 8
f'	4	4 0 D 5
c'	4	4 1 1 C
h'	4	4 0 9 6
g'	4	4 0 B D
e'	4	4 0 E 1
c''	12	C 0 8 D
h'	4	4 0 9 6
h'	4	4 0 9 6
h'	4	4 0 9 6
g'	6	6 0 B D
h'	2	2 0 9 6
a'	4	4 0 A 8
h'	4	4 0 9 6
h'	4	4 0 9 6
h'	4	4 0 9 6
g'	6	6 0 B D
h'	2	2 0 9 6
a'	4	4 0 A 8
h'	4	4 0 9 6
a'	4	4 0 A 8
g'	4	4 0 B D
f'	12	C 0 D 5
Pause	6	6 0 0 1

Hans-Georg Joepgen

Kaufmannskünste für den PET

Dem BASIC-Interpreter des PET 2001 fehlt leider die Fähigkeit, Zahlenkolonnen sauber formatiert und komma-bündig auszugeben. Dennoch: Mit ein wenig Nachhilfe durch ein geeignetes Programm kann man dem PET auf die Sprünge helfen, wie der folgende Beitrag zeigt.

Nächst dem Besuch beim Zahnarzt gehört das Vorbereiten der alljährlichen Einkommenssteuer-Erklärung in des Verfassers Leben zu den übelsten der immer wiederkehrenden Erfahrungen. Besonders lästig: Die Zettelschlacht mit den vielen hundert kleinen Belegen, die im Laufe von zwölf Monaten zusammenkommen; Briefmarken-Quittungen, Ferngespräche, Farbband, dort mal einem Verleger ein Bier spendiert: Pfennigkram zumeist, aber es summiert sich. Als nun der Abgabe-Termin der Steuererklärung immer näher rückte, da beschloß der Autor, diesmal zu probieren, ob ihm nicht sein PET einen Teil der Arbeit abnehmen könne – dies umso mehr, als gerade ein Drucker mit wunderschönem Schriftbild zur Verfügung stand, so schön, daß – wie der Verfasser erwartete – schlechthin jeder Finanzbeamte mit unverbogenem ästhetischen Empfinden gar nicht anders konnte, als eine dergestalt gedruckte Steuererklärung mit Wohlbehagen zu betrachten.

Nun, nach den üblichen Irrwegen, Pannen und Pleiten beim Programmieren hatte der PET kapiert, was von ihm verlangt wurde, und in nicht ganz einer Stunde war danach die Hauptarbeit der Steuererklärung für diesmal erledigt.

Das Programm zeigt nicht nur, wie man mit Hilfe geeigneter Tricks dem PET sauberes Kolonnendrucken und kaufmännisches Runden von Zahlen beibringt, sondern auch, wie der eingebaute Kassettenrekorder zur Protokollführung über jedwede Eingabe eingesetzt werden kann.

Zettelwirtschaft auf Band gebannt

Die Ausgangssituation: Alle Ausgaben-Belege sind, zum Teil mit einem Stichwort zur Notwendigkeit der Ausgabe oder zur Verwendung der gekauften Ware, jeweils auf Einzelblättern aufgeklebt. Dabei kann die Belegzahl pro Blatt zwischen 1 und mehr als einem Dutzend schwanken, je nachdem, ob es sich um formvollendete DIN-A4-Rechnungen oder um kleine Quittungszettelchen handelt, wie sie beispielsweise die Bundespost ihren Kunden aushändigt.

Der Rechner soll nun den jeweiligen Beitragswert all dieser Zettelchen aufaddieren, ein Verzeichnis mit dem jeweils für ein Belegsammelblatt geltenden Summenwert anfertigen und schließlich die Endsumme ausdrucken. Darüber hinaus soll der PET zu Kontrollzwecken und zur Protokollierung die Eingaben auf Band mitschreiben. Schließlich müßte das Programm in der Lage sein, das Protokollband zu lesen und, nun ohne erneute manuelle Eingabe der Belegdaten, ein Duplikat des Verzeichnisses zu schreiben. Für alle diese Einzelanforderungen gilt, daß die Plausibilität der Eingabe überwacht werden soll, daß Flüchtigkeitsfehler beim Tippen ignoriert und schwerwiegendere Bedienfehler durch Schirmmitteilung in Klartext und eine akustische Beschwerde vom Drucker anzuzeigen sind.

Zum Schluß noch eine formale Anforderung an das Programm: Es sollte möglichst klar dokumentiert sein und, soweit möglich, einzelne Programm-Module enthalten, die auch für andere Kaufmanns- und Buchhaltungsprogramme dienen könnten. Wie den Forderungen dieser ‚Pflichtenliste' nachgekommen wurde, das zeigt das Programm-Listing *(Bild 1)*.

Die Weichen stellt ein „Flag"

Nach Laden und Start erkundigt sich das Programm, ob der Operator Daten von Hand einzugeben oder eine Bandaufzeichnung auszuwerten wünscht. Der Rechner verharrt in einer Warteschleife, die er nur nach Eingabe von B für „Bandauswertung" oder von E für „Eingabe" verlassen kann.

Wir betrachten hier zuerst einmal den Fall „Eingabe". Aus der Warteschleife gelangt der Rechner nach Zeile 360. Dort wird zur Sicherheit erst einmal Anweisung zur Schließung des Datenkanals (des „Files") 1 gegeben, das sorgt für klare Verhältnisse, wenn der PET durch eine vorangegangene Programmunterbrechung irritiert und nicht durch einen RUN-Befehl, sondern durch GOTO-Anweisung gestartet wurde. Mit dem folgenden OPEN-Befehl eröffnen wir den Datenkanal 1 neu und definieren ihn als zum Gerät mit der IEC-Bus-Adresse 7 gehörig, dem Drucker. Der letzte Parameter hinter OPEN sagt dem Rechner, daß es sich um einen Ausgabe-Kanal handelt.

Es folgt ein zweiter OPEN-Befehl, der einen Datenkanal mit der Nummer 2 zum eingebauten Kassettenrekorder schaltet: Jetzt kann's losgehen, der PET sendet Signale über den IEC-Bus zum Drucker, der die Adresse des „Steuerpflichtigen" und einen Tabellenkopf druckt. Was der Rechner jetzt weiter tut, hängt vom Inhalt der Variablen AF ab – des „Aufzeichnungs-Flags", das hier im Programm die Weichen stellt und darüber entscheidet, welche Wege der PET geht.

Wir untersuchen hier ja erst einmal die Eingabe. In diesem Fall ist das Flag AF auf Null gesetzt. Nun fordert der PET zur Dateneingabe auf und addiert jedwede Zahl den Konten SS (Gesamtsumme) und BS (Blattsumme) hinzu. Wird ein Pluszeichen eingegeben, so ist das für den Rechner das Signal, den Inhalt des Blattkontos zu formatieren, das

Ergebnis zusammen mit der laufenden Blattzahl auszudrucken und das Blattkonto zu löschen: Jetzt kommt ja ein neues Blatt an die Reihe.

Erkennt der Rechner ein Gleichheitszeichen, so schließt er das zuletzt behandelte Blatt ab, soweit noch nicht geschehen, und meldet das Ergebnis seiner Bemühungen mit dem bedeutungsschweren – und übrigens präzis in der Fachsprache von Finanzbeamten abgefaßten – Satz: „Die Summe der Betriebsausgaben beträgt mithin...", gefolgt vom formatierten Ausdruck der Summe SS aller Blattsummen BS. CLOSE1, CLOSE2: Beide Datenkanäle werden wieder aufgelöst – und Ende.

```
300 PRINTCHR$(147)"ZUSAMMENFASSUNG VON STEUERBELEGEN"
310 PRINT"----------------------------------"
320 PRINT:PRINT:PRINT" +++ EINGABE (E) ODER BANDAUSWERTUNG (B) ?"
330 GETA$:IFA$=""THEN330
340 IFA$="B"THEN990
350 IFA$<>"E"THEN330
360 CLOSE1:OPEN1,7,1:OPEN2,1,1
370 PRINT:PRINT" +++ BITTE ZAHLENEINGABEN UND KOMMANDOS!"
380 B=1:PRINT#1:PRINT#1:PRINT#1
390 PRINT#1,"INNOZENZ MUSTERMAYER"
400 PRINT#1,"STEUERNUMMER 85004-92285"
410 PRINT#1,"ANLAGE ZUR EINKOMMENSTEUER-ERKLAERUNG 1979"
420 PRINT#1:PRINT#1,"AUFSTELLUNG DER BETRIEBSAUSGABEN"
430 PRINT#1,"---------------------------------"
440 IFAFTHENINPUT#2,A$:GOTO460
450 INPUTA$:PRINT#2,A$
460 IF(A$="+")AND(SB<.01)THENGOSUB920:GOTO440
470 IF(A$="=")AND(SB>0)THENGOSUB560:A$="="
480 IFA$="="THENGOSUB620:CLOSE1:CLOSE2:END
490 IFA$="+"THENGOSUB560:GOTO440
500 A=VAL(A$):IFA<.01THENGOSUB920:GOTO440
510 IFSB=0THENGOSUB710
520 SB=SB+A:GOTO440
530 REM
540 REM EINZELBLATT ABSCHLIESSEN
550 REM ------------------------
560 A=SB:GOSUB790:PRINT#1,A$" DM"
570 SS=SS+SB
580 SB=0:B=B+1:RETURN
590 REM
600 REM LISTE ABSCHLIESSEN
610 REM ------------------
620 PRINT#1:PRINT#1:PRINT#1
630 PRINT#1,"   +++ DIE SUMME DER BETRIEBSAUSGABEN
    BETRAEGT MITHIN *";
640 A=SS:GOSUB790:PRINT#1,A$" DM"
650 FORN=1TO68:PRINT#1,"=";:NEXT
660 FORN=1TO10:PRINT#1:NEXT
670 RETURN
680 REM
690 REM BLATTNAMEN-DRUCK
700 REM ----------------
710 B$=STR$(B):IFB<10THENB$=" "+B$
720 PRINT#1,"BELEGSAMMLUNG BLATT-NR.";
730 PRINT#1," "B$" *** SUMME: ";
740 RETURN
750 REM
760 REM FORMATIEREN
770 REM -----------
780 REM
790 A=INT(A*100+.5)/100
800 IFA=0THENA$="      .--":RETURN
810 A$=MID$(STR$(A),2)
820 IFA<1THENA$="0"+A$
830 C$=MID$(STR$(INT(A)),2)
840 IFLEN(C$)<6THENC$=" "+C$:A$=" "+A$:GOTO840
850 IFLEN(A$)=6THENA$=A$+"."
860 IFLEN(A$)=<8THENA$=A$+"0":GOTO860
870 A$=MID$(A$,1,3)+" "+RIGHT$(A$,6)
880 RETURN
890 REM
900 REM FEHLERMELDUNG
910 REM -------------
920 PRINT#1,CHR$(7);
930 PRINT:PRINT"F E H L E R - IGNORIERT"
940 PRINT" +++ WIEDERHOLEN BITTE!"
950 PRINT:RETURN
960 REM
970 REM BANDAUSWERTUNG
980 REM --------------
990 CLOSE1:CLOSE2:OPEN1,7,1:OPEN2,1,0
1000 AF=1:GOTO380
```

Bild 1. Zusammenfassung von Steuerbelegen – eine wahre Hilfe für jedermann. Die Variablen des Programms sind wie folgt belegt: A = Formatierung, B = Blattnummer, N = Schleifenzähler, A$ = Zeichen-Akku, B$ = formatierte Blattzahl, C$ = Kontrollstring, AF = Aufzeichnungs-Flag, BS = Blattsumme, SS = Gesamtsumme. Die Bedienung ist sehr einfach: Zahl/Return gibt einen Summanden ein, Plus/Return schließt eine Seite ab, und Gleich/Return schließt die gesamte Liste ab

Außer dem Verzeichnis der Blattsummen und dem ausgedruckten Satz von der „Summe der Betriebsausgaben" steht nun ein Datenband zur Verfügung, das alle Eingaben enthält: mehr Information als die gedruckte Liste zeigt, die ja nur die Blattsummen zusammenfaßte. Das Band zeigt zusätzlich, aus welchen Summanden sich die Blattsummen jeweils zusammensetzten. Geschehen ist die eigentliche Eingabe-Aufzeichnung in Zeile 450, wo der Rechner sich das oder die Eingabe-Zeichen unter dem Variablen-Namen A$ merkte und sie auf Band schrieb.

Was geschieht nun beim Band-Auswerten? Der Operator beantwortet die Eingangsfrage des Rechners mit dem Betätigen der Taste B (wie „Bandaufzeichnung auswerten"). Darauf springt der Rechner in das Unterprogramm ab Zeile 990, schließt sicherheitshalber die Datenkanäle 1 und 2 und eröffnet sie dann neu, wobei er den Kanal 1 wiederum dem IEC-Bus-Teilnehmer mit der Adresse 4 und somit dem Drucker zuweist, Kanal 2 wird zum Kassettenrekorder durchgeschaltet; nun jedoch nicht zum Schreiben, sondern zum Lesen, wie der dritte OPEN-Parameter „Null" anzeigt.

Das Wichtigste aber: Jetzt wird der „Weichensteller", das Aufzeichnungsflag AF, mit dem Wert 1 belegt. Das Unterprogramm gibt zurück nach Zeile 380, und fortan kümmert sich der Rechner nicht mehr um Tastatur-Eingaben, was für ihn von nun an zählt, sind allein die von Band zurückkommenden Zeichen. Statt des „INPUT" ohne Nummer in Zeile 450, was schlicht Abholen eines Zeichens oder einer Zeichenfolge von der Tastatur bedeutet, ist ab jetzt ein „INPUT#2" in Funktion: Abholung über den Datenkanal 2, den wir dem eingebauten Kassettenrekorder zugewiesen haben. Nun tut der PET seine Pflicht, bis ihm der vom Band stammende Code für ein Gleichheitszeichen das Signal zum Listenabschließen und Schlußmachen gibt. Soweit zum generellen Ablauf: Wie aber sehen die Einzelheiten aus, beispielsweise die Formatierung?

PET-Ausgaben gewinnen Format

Fortran-Rechner und Basic-Maschinen mit besonders komfortablem Interpreter wie der TRS-80 zum Beispiel beherrschen den Umgang mit einer PRINT USING genannten Anweisung,

```
5 REM**PROGRAMM ZUM LESEN DER "STEUERKASSETTE"***
10 OPEN1,7,1:OPEN2,1,0
20 INPUT#2,T$
30 PRINT#1,T$
40 GOTO20
50 CLOSE1:CLOSE2
```

Bild 2. Mini-Programm zum Ausdruck der auf Band gespeicherten Belegliste

die dem Computer sagt, wie er Zahlen bei der Ausgabe anzuordnen habe, damit Komma hübsch unter Komma steht. Dem PET ist derart penibles Bürokratendenken fremd – man muß es ihm erst beibringen. Dies kann beispielsweise über den Zehner-Logarithmus der auszugebenden Zahl erfolgen, wie es der Verfasser im Sonderheft „Hobbycomputer 1" beschrieb – es geht aber auch auf einem mathematisch weniger aufwendigen Weg, mit Hilfe einer Kontroll-Zeichenkette nämlich.

Das Ganze geschieht so: Die zu formatierende Zahl wird mit dem Variablen-Namen A an das Unterprogramm übergeben. Diese Zahl runden wir erst einmal auf volle Pfennig-Beträge. Hat sie danach den Wert Null, setzt das Unterprogramm einen Punkt und zwei Striche als Zeichenkette A-String und gibt zurück.

War das nicht der Fall, liest der Rechner über die MID$- und die STR$-Funktion die Zahl A dergestalt in die Zeichenkette A$ ein, daß auf das linke Kettenglied verzichtet wird. Dies muß sein, weil wir sonst nicht zu klaren Verhältnissen kommen: STR$ gibt beim PET mal eine Leerstelle als Anfangszeichen zurück, mal nicht, je nachdem, wie groß die in einer Zeichenkette umzuwandelnde Zahl war.

Ist A kleiner als 1, setzten wir vor die Zeichenkette in A-String nun eine Null. Damit korrigieren wir die angelsächsische Angewohnheit unseres Rechners, für 0,50 schlicht .5 zu schreiben: Allerdings nur, was die führende Null angeht – die Null nach der 5 bedarf noch gesonderter Behandlung.

Ab Zeile 830 wird's ein wenig verwickelt. Um fortfahren zu können, errichten wir eine Kontroll-Zeichenkette C$ aus dem ganzzahligen Anteil von A. Nun fügen wir solange Leerstellen an den Anfang von A$ und C$ hinzu, bis C$-String genau 6 Zeichen umfaßt. Damit wissen wir: Wenn in A$ bereits ein Punkt als Abbild des Dezimalkommas steht, dann muß es ein siebentes Zeichen der Kette sein – mit Hilfe dieses Wissens können wir in Zeile 850 eventuell fehlende Dezimalkomma-Punkte einfügen.

Der Rest ist wieder einfach: Wenn wir am Schluß der Zeichenkette A$ so lange Nullen anfügen, bis die Zeilenlänge größer als 8 ist, dann haben wir's fast geschafft: In A$ liegt das formatierte Abbild der Zahl A nun bereits vor. Es fehlt nur noch eine Kleinigkeit, die für Leute mit hohen Umsätzen freilich von Bedeutung ist: Das Trennen von Tausender-Kolonnen durch Leerstellen.

In Zeile 870 wird's besorgt: Wir bilden eine neue Zeichenkette, bestehend aus den drei ersten Zeichen der alten, gefolgt von einer Leerstelle, der sodann der Rest der alten Kette folgt. Das reicht für Steuererklärungen mit Betriebskosten bis 999 999 Mark und 99 Pfennige. Wer höhere Betriebsausgaben geltend machen will, der tut's wohl ohnehin nicht mit Hilfe des PET.

In Zeile 880 gibt das Unterprogramm zurück an das aufrufende Programmsegment, dort wird A$ dann ausgedruckt, sauber und ordentlich, Dezimalpunkt unter Dezimalpunkt, gleichgültig, ob's Pfennig-Beträge sind oder Hundertausende: Akkurat, wie's Buchhalter (und Finanzamts-Beamte) nun mal schätzen!

In *Bild 2* findet sich schließlich ein kleines Programm, mit dem die gesamte auf Band gespeicherte Liste auf einem Drucker ausgegeben werden kann.

PET macht Musik

Das im *Bild* aufgelistete Basic-Programm erzeugt an CB2 des User-VIA 6522 im PET/CBM eine vorprogrammierte Musik. Ein Lied kann beliebig oft und auch in unterschiedlichen Oktaven abgespielt werden. Die Informationen über Tonhöhe und -dauer werden in DATA-Statements ab Zeile 1250 abgelegt.

Beim Programmieren eigener Melodien geht man sinnvollerweise wie folgt vor: Zunächst besorgt man sich die Noten des abzuspielenden Stücks (einstimmig genügt). In die DATA-Anweisungen ab Zeile 1250 schreibt man nun abwechselnd die Notennamen und die Tondauer: Eine 1 bedeutet eine Achtelnote, eine 2 eine Viertel-, eine 4 eine halbe, eine 8 eine ganze Note. Pausen werden in der Notenliste mit „P" gekennzeichnet. Halbtöne werden durch ein vorangehendes Doppelkreuz gekennzeichnet (#), z. B. #C entspricht dem Cis. Insgesamt stehen zwei Oktaven zur Verfügung, nämlich von C bis C2.

Falls während des Durchlaufs keine Taste gedrückt wird, wiederholt sich die Melodie ununterbrochen. Die Tasten 1, 2 und 3 ändern die Tonlage, Taste E beendet die Musik. Das Programm enthält ein kleines Melodie-Programmierbeispiel, nämlich die bekannte „Vogelhochzeit".

Willi Gietmann, Willi Billen

```
930 A=59467:B=59466:C=59464
940 POKEA,16:POKEB,20:POKEC,0
950 READT$:REM---TONHOEHE
960 IFT$="ENDE"THEN1060
970 GOSUB2050
980 READT:REM---TONDAUER
990 POKEC,0
1000 FORI=1TO10:NEXTI
1010 POKEC,F
1020 FORI=1TOT
1030 FORW=1TO50:NEXTW
1040 NEXTI
1050 GOTO950
1060 GETT$
1070 IFT$="1"THENPOKEB,15
1080 IFT$="2"THENPOKEB,51
1090 IFT$="3"THENPOKEB,85
1100 IFT$="E"THENPOKEA,0:END
1110 POKEC,0:REM---WIEDERHOLUNG
1120 FORW=1TO500:NEXTW
1130 RESTORE
1140 GOTO950

1290 DATAH,2,D1,2,H,2,D1,2,H,2,C1,2,A,2
1300 DATAC1,2,A,2,H,2,G,2,D1,2,H,2,A,2
1310 DATAD1,2,D1,3,P,4,D,1,D,1,G,2,G,2
1320 DATAG,2,H,1,H,1,A,2,A,2,A,2,C1,1
1330 DATAC1,1,H,2,G,2,A,2,#F,2,G,4,P,8
2040 DATA"ENDE"
2050 IFT$="C"THENF=237
2060 IFT$="#C"THENF=224
2070 IFT$="D"THENF=211
2080 IFT$="#D"THENF=199
2090 IFT$="E"THENF=188
2100 IFT$="F"THENF=177
2110 IFT$="#F"THENF=167
2120 IFT$="G"THENF=157
2130 IFT$="#G"THENF=149
2140 IFT$="A"THENF=140
2150 IFT$="#A"THENF=132
2160 IFT$="H"THENF=124
2170 IFT$="C1"THENF=117
2180 IFT$="#C1"THENF=111
2190 IFT$="D1"THENF=104
2200 IFT$="#D1"THENF=99
2210 IFT$="E1"THENF=93
2220 IFT$="F1"THENF=88
2230 IFT$="#F1"THENF=83
2240 IFT$="G1"THENF=78
2250 IFT$="#G1"THENF=73
2260 IFT$="A1"THENF=69
2270 IFT$="#A1"THENF=65
2280 IFT$="H1"THENF=62
2290 IFT$="C2"THENF=57
2300 IFT$="P"THENF=0
2310 RETURN
```

Dieses Basic-Programm bringt den PET-2001 dazu, die bekannte Melodie „Vogelhochzeit" zu spielen. Das Selbstprogrammieren anderer Stücke ist problemlos möglich. Scheinbar sinnlos im Listing stehende Buchstaben sind Steuerzeichen des PET in Print-Befehlen

Reinhard Wiesemann

Fernschreiber als Drucker für den TRS-80

Baudot-Fernschreiber gehören nach wie vor zu den preiswertesten Ausgabe-Möglichkeiten für Mikrocomputer. Hier wird beschrieben, wie sich auch ohne Maschinenprogramme ein solches Gerät am TRS-80 einsetzen läßt.

Über kurz oder lang wünscht sich jeder Mikrocomputer-Besitzer einen Drucker, z.B. für das Schreiben von Standard-Briefen, wie Bestellungen oder Anforderung von Unterlagen, das Auflisten eines Programmes oder den Ausdruck von Rechenergebnissen. Ausgemusterte Postfernschreiber, die häufig schon für rund 100 DM zu haben sind, eignen sich ganz hervorragend dafür, wenn man hinnimmt, daß sie entweder nur groß oder nur klein schreiben. Für den Anschluß an den TRS-80 ist lediglich noch das seit einiger Zeit erhältliche RS-232-Interface notwendig, das die Parallel/Serienwandlung durchführt. Die dazu notwendige Schaltung zeigt *Bild 1*.

Um Störungen zu vermeiden, sind Computer und Fernschreiber über einen Optokoppler galvanisch voneinander getrennt. Der Transistor T1 muß eine hinreichend hohe Kollektor-Emitter-Spannung vertragen (ca. 70 V) und wird mit D1 vor Schaltspitzen geschützt.

Da der Fernschreiber im Baudot-, der Computer aber im ASCII-Code arbeitet, ist ein kleines Steuerprogramm erforderlich. *Bild 2* zeigt das Listing dieses Unterprogramms. Vor dem ersten Drucken ist ein GOSUB 19800 erforderlich, um den UART-Baustein im Interface vorzubereiten. Der zu schreibende Text wird dann jedesmal in die Variable TE$ gelegt und mit GOSUB 20000 ausgedruckt. Sollen Zahlen ausgegeben werden, so geschieht das mit GOSUB 19900; die Zahl muß vorher in ZA gelegt werden. Da das Programm von sich aus keinen Zeilenvorschub generiert, lassen sich mehrere Zahlen in einer Zeile ausdrucken.

Leider besitzt der Fernschreiber nicht den gleichen Zeichenvorrat wie der Mikrocomputer TRS-80. Aus diesem Grund werden manche Sonderzeichen durch ähnlich aussehende Baudot-Zei-

```
19800 rem sbr vorbereitung zum drucken
19810 dimch(26),nu(30):out+he8,0:out+he9,0:out+hea,+h9c:retu
rn
19900 rem sbr fernschreiberausgabe des inhaltes von za
19910 te?=str?(za)
20000 rem sbr fernspreiberausgabe des textes in te?
20005 restore
20010 data3,25,14,9,1,13,26,20,6,11,15,18,28,12,24,22,23,10,
5,16,7,30,19,29,21,17
20015 data5,255,25,29,255,5,15,18,28,17,12,3,28,29,22,23,19,
1,10,16,21,7,6,24,14,28,15,30,18,25
20020 forch=1to26
20030 readch(ch)
20040 nextch
20042 forch=1to30:readnu(ch):nextch
20050 forle=1tolen(te?)
20060 a=asc(mid?(te?,le,1))
20062 ifle()1thenb=asc(mid?(te?,le-1,1))
20063 if((a)64)and(b(64))or((le=1)and(a)64))thenout+heb,31:f
orn=1to30:nextn
20065 if((a(64)and(b)64))or((le=1)and(a(64))thenout+heb,27:f
orn=1to30:nextn
20070 if(a)64)and(a(91)thena=ch(a-64):goto20130
20080 if(a=128)or(a=32)thena=4:goto20130
20090 if(a)33)and(a(64)thena=nu(a-33):goto20130
20100 if(a=10)or(a=11)or(a=12)or(a=13)thena=8:out+heb,a:forn
=1to30:nextn:a=2:goto20130
20110 a=255
20120 printa
20130 out+heb,a
20133 printmid?(te?,le,1).
20135 forn=1to25:nextn
20140 nextle
20150 return
```

Bild 2. Unterprogramme zum Vorbereiten des UART im Interface und zum Ausdruck von Texten und Zahlen

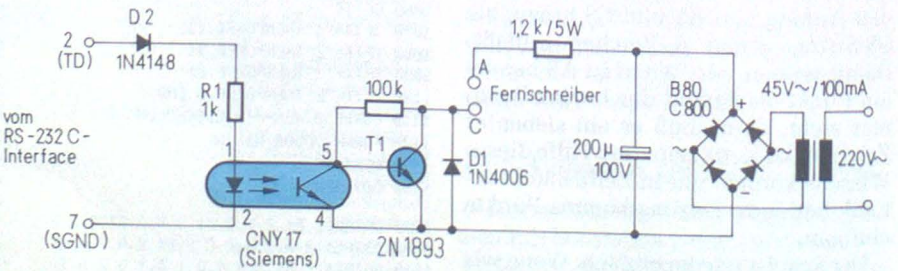


Bild 1. Schaltung zum Ansteuern eines Baudot-Fernschreibers über das RS-232-Interface des Mikrocomputers TRS-80

```
10 rem die zeilen 10 bis 130 demonstrieren, wie das eigentli
che
20 rem programm zur fernschreiber-ansteuerung (zeilen 19800
30 rem bis 20150) benutzt wird.
40 rem  1) vorbereitung vor dem ersten drucken
50 gosub 19800
60 rem  2) text drucken
70 lineinputte?
80 gosub20000
90 rem  3) zahlen drucken
100 inputza
110 gosub19900
120 goto100
```

Bild 3. Aus diesem kurzen Basic-Programm wird deutlich, wie die Unterprogramme in Bild 2 richtig aufgerufen werden

chen ersetzt, an die man sich aber schnell gewöhnt. Statt des Dollar- wird ein Fragezeichen ausgegeben, und statt der Größer- bzw. Kleiner-Zeichen geschlossene bzw. offene Klammern. Eine Verwechslungsgefahr besteht kaum, da das richtige Zeichen immer aus dem Zusammenhang erkennbar ist.

Schnelles Löschen von Programmzeilen beim PET

Das vorliegende Programm dient dazu, Blöcke von Programmzeilen oder auch einzelne Zeilen eines Basic-Programmes zu löschen (*Bild*). Die Vorteile eines Maschinenprogrammes sind hauptsächlich in dessen Schnelligkeit zu sehen, aber auch in der Möglichkeit, es zu laden, während sich das Hauptprogramm bereits im Speicher befindet; dies wäre bei Basic-Programmen nicht ganz so einfach.

Doch nun zum Programm: Die Zeilen 033A-0361 lesen die beiden Zeilennummern ein, zwischen denen ein Programmteil gelöscht werden soll (die beiden Zeilen werden, falls vorhanden, ebenfalls gelöscht). Anschließend werden die zugehörigen Zeilen im Speicher gesucht (0362-038F). Die Nummer der Anfangszeile befindet sich dann in 0050/0051, ihre Speicheradresse in 0054/0055; die Nummer der Endzeile ist in 0052/0053 zu finden, ihre Adresse steht in 0056/0057.

Jetzt wird der hinter der letzten zu löschenden Zeile stehende Programmteil über den zu löschenden Bereich verschoben (0390-03AD). Schließlich werden noch die Basic-Pointer (007A-0087) korrigiert. Außerdem muß auch noch die vor der Zeilennummer jeder verschobenen Zeile stehende Adresse der jeweils nächsten Zeile geändert werden (03AE-03CC).

Hiermit ist der Löschvorgang abgeschlossen; FRE(0) liefert wie gewöhnlich die Anzahl der freien Speicherstellen. Der Programmteil von 03CD-03FF dient dazu, nach dem Einlesen des Delete-Programmes, die durch das Betriebssystem geänderten Pointer (007A-0087) wieder auf die vor dem

```
033A A9 02     LDA #$02
033C 48        PHA
033D 20 36 E2  JSR E236
0340 20 17 CB  JSR CB17
0343 A9 0A     LDA #$0A
0345 85 C9     STA  C9
0347 A9 00     LDA #$00
0349 85 CA     STA  CA
034B 20 A4 CC  JSR CCA4
034E 20 D0 D6  JSR D6D0
0351 68        PLA
0352 AA        TAX
0353 A5 08     LDA  08
0355 95 50     STA  50,X
0357 A5 09     LDA  09
0359 95 51     STA  51,X
035B CA        DEX
035C CA        DEX
035D 8A        TXA
035E 48        PHA
035F 10 DF     BPL 0340
0361 68        PLA
0362 20 22 C5  JSR C522
0365 90 0E     BCC 0375
0367 A0 00     LDY #$00
0369 B1 AE     LDA  (AE),Y
036B 85 56     STA  56
036D C8        INY
036E B1 AE     LDA  (AE),Y
0370 85 57     STA  57
0372 C8        INY
0373 D0 08     BNE 037D
0375 A5 AE     LDA  AE
0377 85 56     STA  56
0379 A5 AF     LDA  AF
037B 85 57     STA  57
037D A5 52     LDA  52
037F 85 08     STA  08
0381 A5 53     LDA  53
0383 85 09     STA  09
0385 20 22 C5  JSR C522
0388 A5 AE     LDA  AE
038A 85 54     STA  54
038C A5 AF     LDA  AF
038E 85 55     STA  55
0390 A0 00     LDY #$00
0392 B1 56     LDA  (56),Y
0394 91 54     STA  (54),Y
0396 E6 54     INC  54
0398 D0 02     BNE 039C
039A E6 55     INC  55
039C E6 56     INC  56
039E D0 02     BNE 03A2
03A0 E6 57     INC  57
03A2 A5 7C     LDA  7C
03A4 C5 56     CMP  56
03A6 D0 EA     BNE 0392
03A8 A5 7D     LDA  7D
03AA C5 57     CMP  57
03AC D0 E4     BNE 0392
03AE A5 54     LDA  54
03B0 85 7C     STA  7C
03B2 85 7E     STA  7E
03B4 85 80     STA  80
03B6 A5 55     LDA  55
03B8 85 7D     STA  7D
03BA 85 7F     STA  7F
03BC 85 81     STA  81
03BE A5 86     LDA  86
03C0 85 82     STA  82
03C2 85 84     STA  84
03C4 A5 87     LDA  87
03C6 85 85     STA  85
03C8 85 83     STA  83
03CA 4C 30 C4  JMP C430
03CD A9 04     LDA #$04
03CF 85 55     STA  55
03D1 A9 00     LDA #$00
03D3 85 54     STA  54
03D5 A0 00     LDY #$00
03D7 A2 03     LDX #$03
03D9 D1 54     CMP  (54),Y
03DB D0 0D     BNE 03EA
03DD CA        DEX
03DE D0 02     BNE 03E2
03E0 F0 10     BEQ 03F2
03E2 E6 54     INC  54
03E4 D0 F3     BNE 03D9
03E6 E6 55     INC  55
03E8 D0 EF     BNE 03D9
03EA E6 54     INC  54
03EC D0 E9     BNE 03D7
03EE E6 55     INC  55
03F0 D0 E5     BNE 03D7
03F2 A5 54     LDA  54
03F4 69 02     ADC #$02
03F6 85 54     STA  54
03F8 A5 55     LDA  55
03FA 69 00     ADC #$00
03FC A5 55     LDA  55
03FE D0 AE     BNE 03AE
```

Disassembliertes Listing des Maschinenprogramms. Es läßt sich z. B. mit dem TIM-Monitorprogramm des CBM eingeben, das mit SYS(1024) aufgerufen wird oder für den PET-2001 auf Kassette lieferbar ist

Einladen bestehenden Werte zu ändern (sonst kann es vorkommen, daß nach bestimmten Operationen ein Reset des Rechners nötig wird).

Zur Handhabung: Unmittelbar nach dem Einlesen werden die Pointer korrigiert. Dies geschieht durch einmaliges Aufrufen der Korrekturroutine mit SYS973. Das Delete-Programm kann jetzt jederzeit mit SYS826 aufgerufen werden. Nach dem Fragezeichen ist dann die erste zu löschende Zeile anzugeben (Return!); entsprechend ist nach dem zweiten Fragezeichen die letzte zu löschende Zeile anzugeben (Return!). Die angegebenen Zeilennummern müssen nicht unbedingt im Hauptprogramm existieren. Das Programm löscht dann alle zwischen diesen beiden Nummern liegenden Zeilen.

Da der Hersteller des Rechners sehr schweigsam in Bezug auf Systemroutinen ist (soll sich doch der Käufer des Rechners selbst daran machen, das Betriebssystem zu „knacken", wenn er auf die „wahnwitzige" Idee kommt, in Maschinensprache zu programmieren), erläutern wir die in diesem Programm verwendeten Routinen etwas näher:

JSR E236 löscht den Bildschirm. JSR CB17 Input, dieser wird als ASCII-String im Input-Puffer (000A-0059) abgelegt. JSR CCA4 wertet einen Term aus, dessen Anfangsadresse in 00C9/00CA steht; hier wird die numerische Eingabe von ASCII nach floating binär konvertiert und in Fließkommaakku 1 abgelegt.

JSR D6D0
Der Inhalt von Fließkommaakku 1 wird nach Integer (16 Bit) konvertiert und in 0008/0009 (Zeilennummer) abgespeichert. JSR C522 sucht nach der in 0008/0009 stehenden Zeilennummer. Wenn diese gefunden wird, Rückkehr mit Carry = 1, Speicheradresse der Zeile in 00AE/00AF; wird die Zeile nicht gefunden, Rückkehr mit Carry = 0, in 00AE/00AF befindet sich dann die Adresse der nächstfolgenden Zeile. JSR C430 korrigiert nach Insert/Delete von Programmzeilen die Adressen der einzelnen Zeilen (vor jeder Zeilennummer steht jeweils die Adresse der nächsten Zeile).

Anzumerken ist noch, daß das Programm voll „relocatible" d.h., es kann ohne Änderungen überall hin verschoben werden.

Das Abspeichern des Maschinenprogramms ist auch ohne TIM-Monitor leicht möglich, nämlich mit der Befehlsfolge:

```
POKE 247,58:POKE 248,3
POKE 229,0:POKE 230,4
SYS (63153)
```

Das Wiedereinlesen des Programms kann dann mit LOAD erfolgen.

Näheres darüber findet sich in FUNKSCHAU 1980, Heft 2, auf Seite 77.

Dieter Smode

µC-Nachrichten

Die von GWK in Aachen entwickelte Basic-Erweiterung für die Computer **AIM-65 und PC-100**, die zusätzliche Befehle für das Öffnen und Schließen von Files, für das Neunumerieren von Zeilen u. a. m. bietet, wird nun auch von Siemens vertrieben; dies geschieht in Form eines EPROM (2716) und eines sechsseitigen Manuals.

□

NAS in München bietet ein EPROM-Programmiergerät für den **Nascom** an, das für die Typen 2708 und 2716 geeignet ist und weniger als 200 DM kostet. Die Software schreibt einen vorwählbaren RAM-Bereich ins EPROM und führt eine Funktionskontrolle durch.

□

Als erste Firma vertreibt Motorola ein dynamisches **64-Kbit-RAM** über seine amerikanischen Distributoren. Allerdings gibt es nach wie vor sehr große Lieferzeiten, und Marketing-Manager Dave Ford äußerte sich sehr unglücklich über Kleinstmengen-Aufträge.

□

Elbatex in Wettingen/Schweiz liefert nun serienmäßig **Magnetblasenspeicher** in handlichen Kassetten mit 64 Kbit und 256 Kbit formatierter Speicherkapazität. Die nichtflüchtigen Speicher besitzen eine Zugriffszeit von typ. 0,37 s auf eine beliebige Schleife. Die „Bubble Cassette" enthält auch den notwendigen Controller für eine Floppy-ähnliche Organisation des Zugriffs.

□

Was Commodore selbst nicht bietet, holen andere nach: Die Münchner Firma Schießl & Partner („Softwareverbund Mikrocomputer") brachte eine Loseblatt-Sammlung als Nachschlagewerk für die **Programmierung der CBM-Computer** heraus. Die Grundinformation umfaßt 155 Seiten. Auf die Programmierung in Assemblersprache unter Ausnutzung des ROM-Betriebssystems und der Floppy wird ausführlich eingegangen. Sogar ein kommentiertes ROM-Listing ist (für knapp 100 DM) von SM zu haben.

□

Entgegen den bisherigen Verlautbarungen von Siemens scheint es offenbar doch einen Weg zu geben, beim Tischcomputer **PC-1000** (s. Heft 11/1980) auf Maschinensprache-Ebene herabzusteigen. Übrigens vertreibt Siemens das Gerät nicht nur selbst; auch von der Aachener Firma GWK kann man den PC-1000 ab September beziehen.

□

Das Berliner Software-Haus **ASM** (Anwender-Software Minhoff & Co.) feiert in diesem Jahr sein zehnjähriges Bestehen. Seit Mitte Mai besitzt Minhoff in der Berliner Bundesallee 160 auch einen Computershop (CSM). Dabei wird besonderer Wert auf eine herstellerunabhängige Beratung gelegt.

Berichtigung

ZETBUG – ein komfortabler Z-80-Monitor

FUNKSCHAU 1980, Heft 11, Seite 101

Wegen eines Fehlers in der TRS-80-Betriebssoftware muß der Befehl E5 (Push HL) von 4400 nach 43EF vorgezogen werden, um ein Fehlverhalten beim M-Kommando zu vermeiden: Manchmal würde fälschlich ROM ERROR ausgedruckt werden. Richtig muß es daher heißen:

```
43EF: E5 CD E1 43 20 23 20 00 CD B8 43
CD C5 43 FE 0D 20 F6 2A...
```

Dipl.-Ing. Karl-Heinz Zeller

Netzwerkanalyse mit dem TI-59

Das Übertragungsmaß elektrischer Netzwerke läßt sich stets durch eine gebrochene rationale Funktion ausdrücken, deren Auswertung hinsichtlich Amplituden- und Phasengang sowie Signallaufzeit auf einfache Weise mit einem programmierbaren Taschenrechner möglich ist.

Das Übertragungsverhalten eines Netzwerks kann – sofern nur konzentrierte Bauelemente verwendet werden, der Skin-Effekt vernachlässigt werden kann und die Schaltung keine Spalt-Tiefpässe oder Abtastfunktionen enthält – durch eine Gleichung folgender Form dargestellt werden:

$$S = \frac{A_o + A_1 p + A_2 p^2 + ... + A_n p^n}{B_o + B_1 p + B_2 p^2 + ... + B_m p^m}$$

Das Programm für den programmierbaren Rechner TI-59 und den Drucker PC-100 *(Bild 1)* verarbeitet Übertragungsfunktionen mit $n + m \leqq 23$, wodurch sich auch komplizierte Regelkreise oder Filterschaltungen durchrechnen lassen. Es berechnet für eine vorzugebende Reihe von Frequenzen Verstärkung bzw. Dämpfung (auch in dB), die Phasendrehung zwischen Ein- und Ausgang sowie die Signallaufzeit. Die Speicher *(Bild 2)* enthalten die Druckertexte.

Die Bedienung des Rechners

Zunächst ist anhand des Schaltbildes die Übertragungsfunktion aufzustellen und unter Verwendung der Grundeinheit (Ohm, Henry, Farad) die Koeffizienten zu berechnen. Dann liest man das Programm ein (Blöcke 1, 2, 3); die Speicherbereichsverteilung ist dabei normal. Der Programmstart erfolgt mit RST R/S. Nun können nacheinander die Koeffizienten des Zählers (ab A_o) eingegeben werden. Jede Eingabe ist mit R/S abzuschließen. Unmittelbar vor oder während der Eingabe des letzten Koeffizienten (A_n) muß man Flag 0 setzen: 2nd Stflg 0 R/S.

Beginnend mit B_o gibt man nun ebenso die Nennerkoeffizienten ein; auch hier ist vor dem letzten Koeffizienten Flag 0 zu setzen. Dann folgt die Startfrequenz in der Einheit Hz. Ist sie Null, so wird automatisch ein lineares, andernfalls ein logarithmisches Frequenzraster gewählt. Nach der anschließenden Eingabe der Endfrequenz druckt der Rechner dementsprechend LIN oder LOG aus.

Um Papier und Rechenzeit einzusparen, kann durch Setzen bestimmter Flags die Berechnung und der Ausdruck uninteressanter Ergebnisse übersprungen werden:

Betrag	Flag 1
Betrag in dB	Flag 2
Phasendrehung, Laufzeit	Flag 3
Signallaufzeit	Flag 4

Im Modus LIN muß man nun noch die Frequenzschrittweite, bei LOG die Zahl der Frequenzschritte pro Dekade eingeben. Die jetzt folgende Rechenzeit ist von der Zahl der Koeffizienten abhängig und kann mehrere Minuten betragen.

Will man nach dem Ablauf des Programms einen anderen Frequenzbereich mit dem gleichen Netzwerk berechnen lassen, so kann man die Neueingabe der Koeffizienten mit der Programmadreßtaste A umgehen.

Fehlermeldungen

Wird die maximale Anzahl der Koeffizienten (25 bzw. $n + m = 23$) überschritten, so erfolgt der Ausdruck „FEHLER!" Das Programm rechnet trotzdem bis maximal 40 Koeffizienten richtig, allerdings wird dann der Textspeicher überschrieben.

Die Übertragungsfunktion kann bei Vernachlässigung von Bauelemente-Verlusten oder Oszillatorschaltungen Nullstellen und Pole (Unendlichkeitsstellen) enthalten. Fällt ein Frequenzwert auf eine solche Stelle, so erfolgt statt eines Zahlenwertes der Ausdruck „*****", und das Programm fährt mit dem nächsten Frequenzwert fort.

Der Winkelbereich des TI-59 reicht von – 90° bis + 270°. Diese Einschränkung kann in bestimmten Grenzfällen zu einem falschen Wert für die Signallaufzeit führen. Das Programm erkennt dies aber rechtzeitig und druckt wiederum „*****" statt der Laufzeit.

Einfaches Rechenbeispiel

Die in *Bild 3* wiedergegebene Tiefpaß-Schaltung soll als Rechenbeispiel dienen. Die Übertragungsfunktion S ist der Quotient von Aus- und Eingangsgröße:

$$S = \frac{\text{Ausgangsgröße}}{\text{Eingangsgröße}} = \frac{U_a}{U_e}$$

Daß S normalerweise – so auch hier – komplex ist, braucht uns dank der bequemen Abkürzung p nicht weiter zu kümmern. Wir müssen nur für den Widerstand einer Induktivität $p \cdot L$ und für den eines Kondensators $1/p \cdot C$ schreiben. Nach der bekannten Spannungsteilerformel

$$\frac{U_a}{U_e} = \frac{R_2}{R_1 + R_2}$$

erhalten wir für den Spannungsteiler aus Widerstand R und Kondensator C:

$$S = \frac{U_a}{U_e} = \frac{\frac{1}{p \cdot C}}{R + \frac{1}{p \cdot C}}$$

Diese Gleichung müssen wir nun nur noch in die gebrochen-rationale Form bringen, wie sie unser Programm verlangt. Dazu wird der Bruch mit $p \cdot C$ erweitert:

$$S = \frac{1}{p \cdot R \cdot C + 1}$$

Die Koeffizienten sind dann: $A_o = 1$, $B_o = 1$, $B_1 = R \cdot C$; alle anderen sind

```
000  69 OP
001  00  00
002  08  8
003  32 X:T
004  43 RCL
005  59  59
006  42 STO
007  00  00
008  02  2
009  00  0
010  42 STO
011  01  01
012  71 SBR
013  02  02
014  66  66
015  43 RCL
016  02  02
017  42 STO
018  10  10
019  43 RCL
020  58  58
021  42 STO
022  00  00
023  22 INV
024  86 STF
025  00  00
026  71 SBR
027  02  02
028  66  66
029  43 RCL
030  02  02
031  42 STO
032  11  11
033  76 LBL
034  11  A
035  69 OP
036  00  00
037  43 RCL
038  57  57
039  69 OP
040  04  04
041  91 R/S
042  69 OP
043  06  06
044  42 STO
045  12  12
046  32 X:T
047  25 CLR
048  22 INV
049  77  GE
050  00  00
051  55  55
052  22 INV
053  86 STF
054  00  00
055  43 RCL
056  56  56
057  69 OP
058  04  04
059  91 R/S
060  69 OP
061  06  06
062  42 STO
063  13  13
064  43 RCL
065  45  45
066  87 IFF
067  00  00
068  00  00
069  72  72
070  43 RCL
071  46  46
072  69 OP
073  04  04
074  69 OP
075  05  05
076  91 R/S
077  99 PRT
078  22 INV
079  97 IFF
080  00  00
081  00  00
082  86  86
083  35 1/X
084  22 INV
085  28 LOG
086  42 STO
087  16  16
088  98 ADV
089  43 RCL
090  51  51
091  69 OP
092  04  04
093  43 RCL
094  12  12
095  57 ENG
096  69 OP
097  06  06
098  65  ×
099  02  2
100  65  ×
101  89  π
102  95  =
103  42 STO
104  14  14
105  42 STO
106  15  15
107  71 SBR
108  03  03
109  79  79
110  87 IFF
111  01  01
112  01  01
113  22  22
114  43 RCL
115  55  55
116  69 OP
117  04  04
118  43 RCL
119  08  08
120  69 OP
121  06  06
122  87 IFF
123  02  02
124  01  01
125  43  43
126  43 RCL
127  08  08
128  28 LOG
129  65  ×
130  02  2
131  00  0
132  95  =
133  32 X:T
134  43 RCL
135  54  54
136  69 OP
137  04  04
138  32 X:T
139  69 OP
140  06  06
141  87 IFF
142  03  03
143  02  02
144  36  36
145  25 CLR
146  32 X:T
147  43 RCL
148  12  12
149  67  EQ
150  03  03
151  47  47
152  43 RCL
153  53  53
154  69 OP
155  04  04
156  43 RCL
157  09  09
158  69 OP
159  06  06
160  87 IFF
161  04  04
162  02  02
163  36  36
164  43 RCL
165  15  15
166  65  ×
167  93  .
168  09  9
169  09  9
170  95  =
171  42 STO
172  14  14
173  71 SBR
174  03  03
175  79  79
176  43 RCL
177  09  09
178  42 STO
179  19  19
180  43 RCL
181  15  15
182  65  ×
183  01  1
184  93  .
185  00  0
186  01  1
187  95  =
188  42 STO
189  14  14
190  71 SBR
191  03  03
192  79  79
193  25 CLR
194  32 X:T
195  43 RCL
196  19  19
197  77  GE
198  02  02
199  11  11
200  43 RCL
201  09  09
202  32 X:T
203  43 RCL
204  47  47
205  77  GE
206  02  02
207  11  11
208  61 GTO
209  03  03
210  47  47
211  43 RCL
212  19  19
213  75  -
214  43 RCL
215  09  09
216  95  =
217  55  ÷
218  07  7
219  93  .
220  02  2
221  55  ÷
222  43 RCL
223  12  12
224  95  =
225  42 STO
226  07  07
227  43 RCL
228  52  52
229  69 OP
230  04  04
231  43 RCL
232  07  07
233  57 ENG
234  69 OP
235  06  06
236  43 RCL
237  16  16
238  87 IFF
239  00  00
240  02  02
241  47  47
242  44 SUM
243  12  12
244  61 GTO
245  02  02
246  49  49
247  49 PRD
248  12  12
249  43 RCL
250  12  12
251  32 X:T
252  43 RCL
253  13  13
254  22 INV
255  77  GE
256  02  02
257  61  61
258  61 GTO
259  00  00
260  88  88
261  98 ADV
262  98 ADV
263  98 ADV
264  98 ADV
265  91 R/S
266  25 CLR
267  42 STO
268  02  02
269  43 RCL
270  00  00
271  69 OP
272  04  04
273  91 R/S
274  69 OP
275  06  06
276  72 ST*
277  01  01
278  43 RCL
279  02  02
280  67  EQ
281  03  03
282  24  24
283  75  -
284  01  1
285  95  =
286  67  EQ
287  03  03
288  29  29
289  75  -
290  09  9
291  95  =
292  67  EQ
293  03  03
294  24  24
295  75  -
296  01  1
297  95  =
298  67  EQ
299  03  03
300  41  41
301  76 LBL
302  22 INV
303  69 OP
304  20  20
305  69 OP
306  21  21
307  43 RCL
308  01  01
309  75  -
310  03  3
311  07  7
312  95  =
313  67  EQ
314  03  03
315  60  60
316  69 OP
317  22  22
318  22 INV
319  87 IFF
320  00  00
321  02  02
322  69  69
323  92 RTN
324  02  2
325  44 SUM
326  00  00
327  61 GTO
328  22 INV
329  01  1
330  00  0
331  22 INV
332  44 SUM
333  00  00
334  01  1
335  00  0
336  00  0
337  49 PRD
338  00  00
339  61 GTO
340  22 INV
341  08  8
342  08  8
343  44 SUM
344  00  00
345  61 GTO
346  22 INV
347  69 OP
348  00  00
349  22 INV
350  57 ENG
351  43 RCL
352  48  48
353  69 OP
354  03  03
355  69 OP
356  05  05
357  61 GTO
358  02  02
359  36  36
360  69 OP
361  00  00
362  22 INV
363  57 ENG
364  22 INV
365  52 EE
366  43 RCL
367  49  49
368  69 OP
369  01  01
370  43 RCL
371  50  50
372  69 OP
373  02  02
374  69 OP
375  05  05
376  61 GTO
377  03  03
378  16  16
379  43 RCL
380  11  11
381  42 STO
382  00  00
383  85  +
384  43 RCL
385  10  10
386  85  +
387  01  1
388  09  9
389  95  =
390  42 STO
391  05  05
392  71 SBR
393  04  04
394  25  25
395  69 OP
396  35  35
397  43 RCL
398  10  10
399  42 STO
400  00  00
401  71 SBR
402  04  04
403  25  25
404  43 RCL
405  17  17
406  42 STO
407  03  03
408  43 RCL
409  18  18
410  42 STO
411  04  04
412  36 PGM
413  04  04
414  18 C'
415  32 X:T
416  22 INV
417  37 P/R
418  42 STO
419  09  09
420  32 X:T
421  42 STO
422  08  08
423  57 ENG
424  92 RTN
425  25 CLR
426  48 EXC
427  01  01
428  42 STO
429  17  17
430  25 CLR
431  48 EXC
432  02  02
433  42 STO
434  18  18
435  73 RC*
436  05  05
437  42 STO
438  03  03
439  25 CLR
440  42 STO
441  04  04
442  36 PGM
443  04  04
444  12  B
445  22 INV
446  97 DSZ
447  00  00
448  04  04
449  65  65
450  43 RCL
451  14  14
452  42 STO
453  04  04
454  25 CLR
455  42 STO
456  03  03
457  36 PGM
458  04  04
459  13  C
460  69 OP
461  35  35
462  61 GTO
463  04  04
464  35  35
465  22 INV
466  37 P/R
467  25 CLR
468  67  EQ
469  03  03
470  47  47
471  92 RTN
```

Bild 1. Vollständiges Programmlisting für die Netzwerkanalyse mit dem Rechner TI-59 nebst zugehörigem Drucker PC-100

```
 273222.      45
 272431.      46
     180.     47
5151515151.   48
  211723.     49
2717350073.   50
 21351734.    51
 27214637.    52
 33231336.    53
  1614.       54
 14173735.    55
  2132.       56
  2141.       57
  1401.       58
  1301.       59
```

Bild 2. Die Speicherregister enthalten die auszugebenden Texte für den Drucker

```
     1. 00    A0
     1. 00    B0
   10. -06    B1
     1. 03    F0
   100. 03    F0
               LOG
    10. 00

     1. 03 FREQ
-17.111504-03   DB
-3.5952738 00 PHAS

 1.2589254 03 FREQ
-27.088746-03   DB
-4.5227144 00 PHAS
```

Bild 4. Eingabeprotokoll zu Bild 3 und die beiden ersten Frequenzwerte

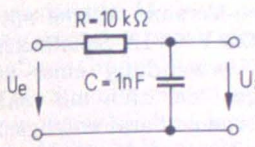


Bild 3. Einfache Beispielschaltung: Ein Tiefpaß

Null. Die Werte der Bauelemente setzen wir in Grundeinheiten ein: $B_1 = 10000 \cdot 10^{-9} = 10^{-5}$. Wir laden das Programm, starten es mit RST R/S und setzen auch Flag 0, da der erste Zählerkoeffizent gleich der letzte ist. Dann ist A_0 (= 1) einzugeben und R/S zu drücken. Wenn die Anzeige wieder erscheint, kann man B_0 (= 1) eingeben, R/S drücken, Flag 0 setzen und B_1 (= 1 EE +/– 5), gefolgt von R/S, eintippen. Am besten sehen wir uns den Frequenzbereich 1...100 kHz an; uns interessieren nur der Betrag in dB und die Phasenverschiebung für 10 Frequenzschritte je Dekade. Also geben wir die Startfrequenz 1000, R/S und die Endfrequenz 100000, R/S ein. Dann setzen wir die Flags 1 und 4 und geben 10 (Zahl der Schritte pro Dekade) und R/S ein. Bild 4 zeigt ein Protokoll der Eingabe und die beiden ersten Frequenzschritte.

RTTY mit Sprachsynthese

Seit dem 1. Juni, dem Inkrafttreten neuer Amateurfunk-Bestimmungen, ist es erforderlich, zu Beginn und Ende jeder Funkverbindung sein Rufzeichen auch dann in Sprache oder Morsen zu nennen, wenn eine Sonderbetriebsart wie etwa Funkfernschreiben (RTTY) verwendet wird. Da Inhaber der C-Lizenz nun zwar RTTY-, nicht aber Morse-Sendungen ausstrahlen dürfen, müssen sie ihr Rufzeichen in Sprache senden. Zwecks erhöhtem Komfort bringen wir nachfolgend eine Erweiterung des 1979 veröffentlichten RTTY-Programms für die Computer AIM-65 und PC-100, das dies automatisch per Sprachsynthese tut.

Für das Hinzufügen der Sprachsynthese ist kein weiterer Hardware-Aufwand nötig: Es genügt, das Programmlisting im *Bild* mit dem RTTY-Programm [1, 2] zu kombinieren.

Aus Platzgründen wurde hier auf den Abdruck eines kompletten Assembler-Listings der Sprachsynthese verzichtet. Das Prinzip wird jedoch zusammen mit dem Quellencode in dem im Oktober erscheinenden FUNKSCHAU-Sonderheft „Mikrocomputer-Anwendungen" näher erläutert [3].

Hat man RTTY- und Sprachsynthese-Programm eingegeben, so ist wie folgt vorzugehen: Man nimmt mit dem Kassettenrecorder den gewünschten Text auf; dabei sollte man, um Nebengeräusche möglichst zu unterdrücken, ein externes Mikrofon verwenden. Dieser Text kann z. B. lauten: „Hier ist DC 1 Yankee Bravo." Eine winzige Hardware-Änderung auf der AIM-Platine sorgt dafür, daß dieser Text mit dem Kassetten-Interface digital in den Speicher geladen werden kann, hier ab Adresse 049D: Parallel zu R 18 (5,1 MΩ) ist ein 390-kΩ-Widerstand zu legen. Das Einlesen von Programmen wird davon nicht beeinflußt.

Jetzt kann man den auf Kassette gesprochenen Text durch Drücken der Taste F2 in den Speicher übernehmen (049D...0DFF). Ein Druck auf F3 (stets vom Monitorprogramm aus gedrückt) gibt den Text an PB7 direkt als Nf-Signal wieder aus. Das Ganze hört sich zwar etwas rauh und komprimiert an, die Verständlichkeit ist aber einwandfrei, und sogar die Stimme des Sprechers ist wiederzuerkennen. Es lohnt sich, mit der Lautstärke-Einstellung des Recorders einige Versuche zu unternehmen, um die beste Qualität zu erreichen.

Sollte sich herausstellen, daß die zur Verfügung stehende Aufnahmezeit nicht ausreicht, so kann man die Page-Grenze des Sprachspeichers an der Adresse 0487 (jetzt 0E) auf max. 10 (4-K-AIM) erhöhen; dabei ergibt sich eine Sprechzeit von rund drei Sekunden, was für die Rufzeichennennung völlig ausreicht.

Alles bis hier besprochene funktioniert auch ohne das RTTY-Programm und kann daher auch für andere Sprachsynthese-Aufgaben verwendet werden.

Das RTTY-Empfangsprogramm wird mit der Taste F1 vom Monitorprogramm aus gestartet; mit Taste S erfolgt wie gewohnt die Umschaltung in den Sendemodus – und jetzt passiert's: Am Anfang der Fernschreibsendung wird

```
[K]*=02FA
/02
 02FA 20 JSR 0449
 02FD 4C JMP 0000

[K]*=00A7
/02
 00A7 20 JSR 0449
 00AA 4C JMP 0200

[K]*=000E
/
 000E 4C JMP 00A7

[M]=0400 A9 37 8D 02 A8 A9 EE 8D 0C A8 20 89 04 E6 FD F0
[ ] 0410 1E A2 08 48 20 92 04 AD 00 A8 0A 68 6A CA D0 F3
[ ] 0420 C9 55 D0 01 0A C5 FC F0 E4 85 FC A4 FD F0 12 A9
[ ] 0430 55 81 FE 20 7E 04 A5 FD 81 FE 86 FD 20 7E 04 A5
[ ] 0440 FC 81 FE 20 7E 04 D0 C9 60 A9 80 8D 02 A0 20 89
[ ] 0450 04 A2 00 A1 FE C9 55 F0 06 85 FC A9 01 D0 05 20
[ ] 0460 7E 04 A1 FE 85 FD A2 08 A5 FC 20 92 04 6A 6E 00
[ ] 0470 A0 CA D0 F6 C6 FD D0 EE 20 7E 04 D0 D6 60 E6 FE
[ ] 0480 D0 02 E6 FF A5 FF C9 0E 60 A9 9D 85 FE A9 04 85
[ ] 0490 FF 60 2C 85 A4 10 FB A0 60 8C 94 A4 60 55 01 FF
[K]*=010C
/03
 010C 4C JMP 0000
 010F 4C JMP 0400
 0112 4C JMP 0449
```

Nur ein relativ kleines Zusatzprogramm ist nötig, um selbst gesprochene Texte digital im RAM des AIM-65 abzulegen und zu Beginn und Ende jeder Funkfernschreib-Sendung „synthetisch" wieder auszugeben, um den Postbestimmungen Genüge zu tun

automatisch das vorher gespeicherte Rufzeichen in Sprache ausgegeben, ebenso wenn man mit F2 vom Sende- in den Empfangsmodus zurückschaltet. F1 gestattet bei laufendem Sendeprogramm auch den Abruf eines Standardtextes.

Speichert man alle Programmteile auf Kassette ab, so sind folgende Adressenbereiche einzugeben: 0000...00AC; 010C...0114; 0170...01E0 (Standardtext), 0200...02FF; 0400...0DFF (oder 0400...0FFF, je nach Sprachdauer, s. o.).

Herwig Feichtinger

Literatur

[1] AIM-65 empfängt RTTY. FUNKSCHAU 1979, Heft 15, S. 884.

[2] AIM-65 sendet RTTY. FUNKSCHAU 1979, Heft 16, S. 953.

[3] Sprachsynthese mit dem AIM-65. Sonderheft „Mikrocomputer-Anwendungen" der FUNKSCHAU, Oktober 1980.

Apple-III ab Oktober

Die US-Firma Apple Co. wird das neue Modell Apple-III ab Oktober 1980 auch nach Europa liefern (*Bild 1*). Der „neue Apple" arbeitet wie sein Vorgänger Apple-II mit dem Mikroprozessor 6502, aber dank 2 MHz Taktfrequenz doppelt so schnell. Ein Floppy-Disk-Laufwerk ist bereits eingebaut, und zusammen mit einem Schwarzweiß-Monitor kostet das System in den USA rund 4300 Dollar.

Bild 2. Der Benutzer kann nicht nur Zeichen auf dem Bildschirm selbst definieren (z. B. die deutschen Umlaute), sondern auch die hochauflösende Grafik des Apple-III für Schriften verwenden

Damit drängt Apple eindeutig – ähnlich wie Siemens mit dem PC-100 (s. Heft 11/1980) – in den professionellen Markt. Dem trug man auch hardwareseitig Rechnung: Der Apple-III kann mit bis zu 128 KByte Arbeitsspeicher ausgerüstet werden. Da ein Mikroprozessor mit 16 Adressenleitungen nur 64 KByte adressieren kann, sorgt eine spezielle Adressenlogik für das „Unmögliche". Für 6502-Kenner: Auch ein Multi-User-Betrieb ist möglich, da Zero-Page und Stack umgeschaltet werden können. Ferner ist eine batteriegepufferte Echtzeituhr vorhanden.

Die Bildschirmausgabe kann farbig oder schwarzweiß in 24 Zeilen zu je 40/80 Zeichen (umschaltbar) erfolgen, und auch die Darstellung hochauflösender Grafiken (*Bild 2*) ist möglich. Basic, Pascal oder Fortran-Compiler/Interpreter sind nicht in ROMs untergebracht, sondern werden mit einem Bootstrap-Ladeprogramm, das die einzige ROM-Software darstellt, in den RAM-Arbeitsspeicher geladen. Da man auch die Apple-II-ROM-Software ins RAM laden kann, ist es sogar möglich, Apple-II-Maschinenprogramme völlig unverändert auf dem Apple-III laufen zu lassen!

Die Europa-Version arbeitet mit einem 220-V/50-Hz-Schaltnetzteil. Durch die Verwendung eines RGB-Farbausganges, der sich mit zahlreichen modernen Farbfernsehgeräten verbinden läßt, wird das Problem der Farbfernsehnorm (NTSC/PAL) umgangen.

Fe.

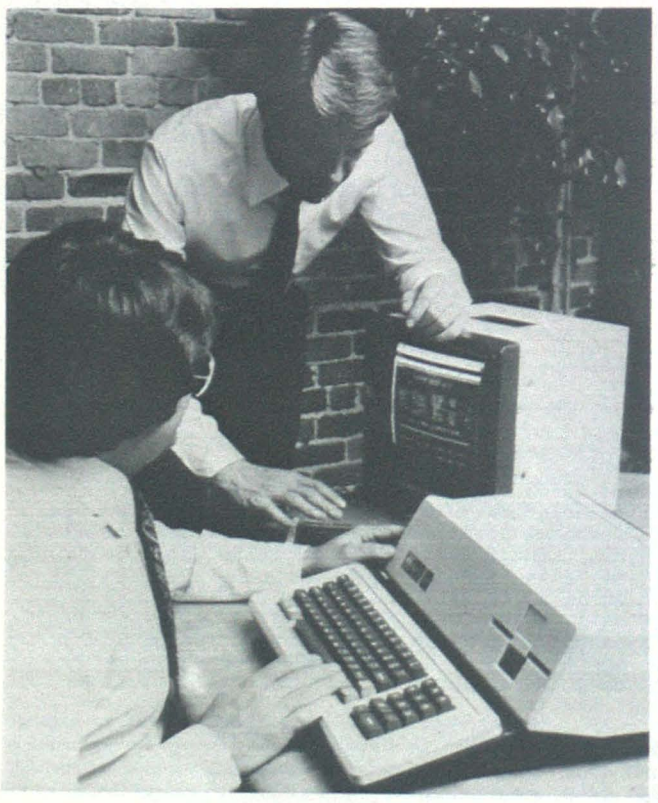


Bild 1. Gegenüber dem Vorläufer Apple-II ist der Tischcomputer Apple-III floppy-disk-orientiert: Im eingebauten Festwertspeicher (ROM) steht nur ein kleines Floppy-Ladeprogramm, und die gesamte übrige Systemsoftware wird von der Floppy-Disk in den Arbeitsspeicher (RAM) geladen

µC-Nachrichten

Das Holzkirchener Software-Haus Redysoft vertreibt nun auch Programme der US-Firma Microsoft, so etwa einen Basic-Compiler für TRS-80, Modell II (d. h. für das neue Business-System von Tandy), ebenso für Modell I. Für den Apple-II sind sogar Basic-Compiler, Cobol und Fortran zu haben. Eine Z-80-Zusatzkarte für den Apple-II, mit der die Verwendung von CP/M-Software möglich wird, ist ebenfalls zu haben.

□

Unter dem Titel **„Basic lernen mit dem PC-100"** brachte Siemens ein 78 Seiten umfassendes Büchlein im A5-Format heraus, das wesentlich instruktiver als vergleichbare Personal-Computer-Handbücher ist. Es stellt alle Basic-Befehlsworte und Systemkommandos ausführlich mit Anwendungsbeispielen dar und sei auch AIM-65-Besitzern mit Basic-ROMs empfohlen.

□

Von Intel war zu hören, daß noch 1981 mit der Vorstellung eines **32-bit-Mikroprozessors** zu rechnen sei. Zwar verspricht man sich von solchen „Komfort-Prozessoren" wegen des enormen Hardware- und Testaufwandes und der damit verbundenen Kosten nur einen relativ geringen Marktanteil, erwartet aber ein Eindringen dieser Technik in größere EDV-Systeme.

PET als Digitalvoltmeter

Hier wird eine einfache Hardware-Erweiterung für Mikrocomputer beschrieben, mit der sich Spannungen messen und weiterverarbeiten lassen. Für den PET-2001 findet sich ein Programm zur Darstellung der Meßwerte in Großbuchstaben.

Das A/D-Wandler-IC

Mikrocomputer können vielfach zur automatisierten Messung und Überwachung analoger Größen eingesetzt werden. Diese Größen liegen entweder direkt als Spannungswert vor oder können in einen solchen konvertiert werden (z. B. Strom, Widerstand, Temperatur usw.)

Das IC MC 14433 von Motorola eignet sich recht gut für unsere Zwecke, weil es über BCD-Ausgänge verfügt. Es arbeitet nach dem Zweiflanken-Verfahren und mit automatischer Polaritätsumschaltung. Der Stromverbrauch ist dank der CMOS-Technologie recht gering. Die BCD-Ausgänge werden gemultiplext, wobei die logischen Zustände der Digit-Select-Ausgänge angeben, welche Stelle gerade angesprochen wird (*Bild 1*).

Das IC benötigt zwei Versorgungsspannungen, deren Gesamtwert 18 V nicht überschreiten darf. Die interne Taktfrequenz wird mit einem externen Widerstand eingestellt, wobei die Meßzeit (inklusive automatischem Nullabgleich) 16 400 interne Taktzyklen beträgt.

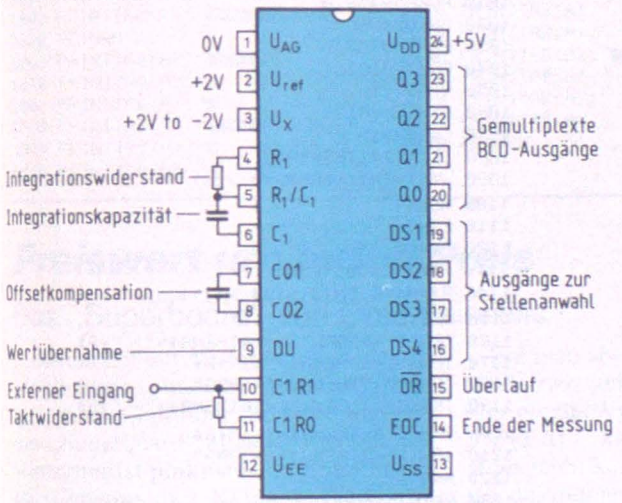


Bild 1. Anschlußbelegung des Digitalvoltmeter-IC MC 14433, das sich wegen seiner BCD-Ausgänge ideal für Mikrocomputer-Anwendungen eignet

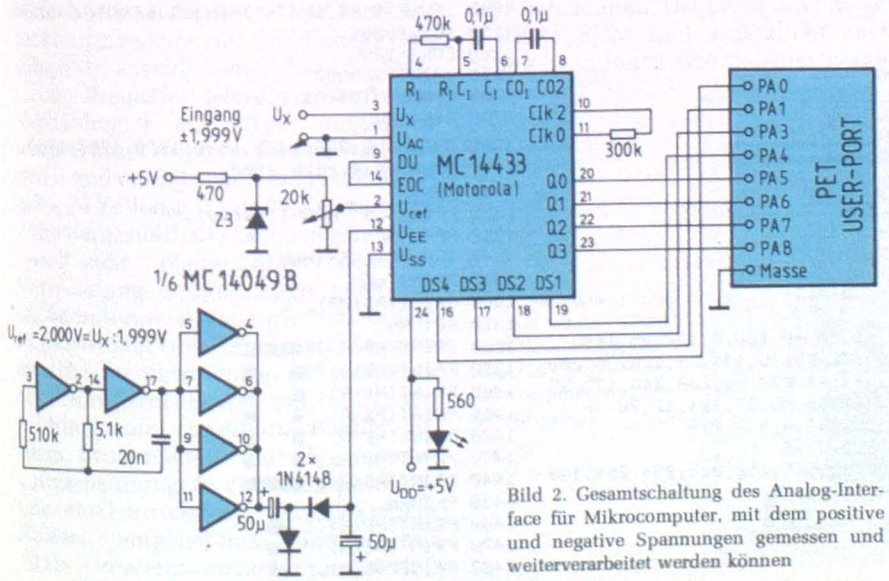


Bild 2. Gesamtschaltung des Analog-Interface für Mikrocomputer, mit dem positive und negative Spannungen gemessen und weiterverarbeitet werden können

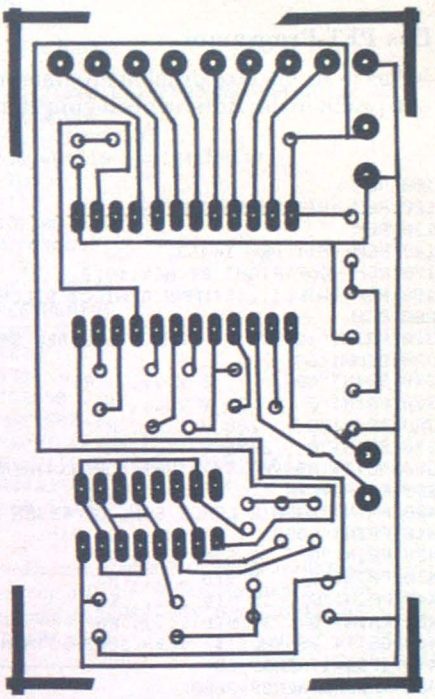


Bild 3. Layout der einseitig kupferkaschierten Platine

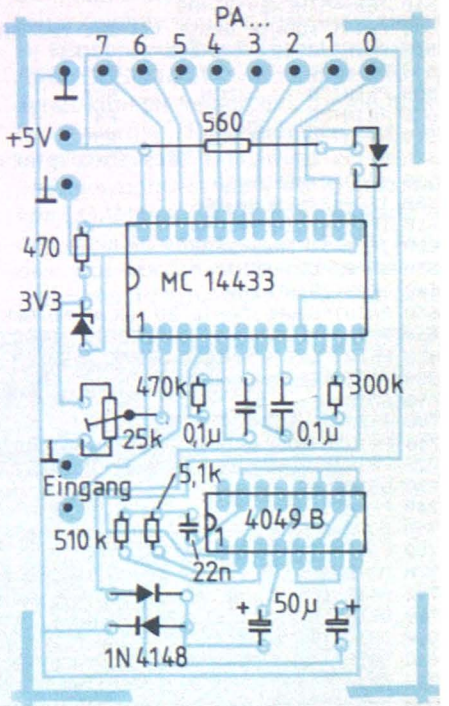


Bild 4. Bestückungsplan der Digitalvoltmeter-Platine. Die Ausgänge PA 0...7 sind mit den entsprechenden User-Port-Anschlüssen des Mikrocomputers zu verbinden

Bild 2 zeigt die Schaltung eines damit aufgebauten Spannungsmesser-Zusatzes für Mikrocomputer. Ein 4049 dient zur Erzeugung der negativen Versorgungsspannung, so daß keine besondere Stromversorgung erforderlich ist. *Bild 3* zeigt ein Platinenlayout, auf dem, wie in *Bild 4* zu sehen ist, alle Bauelemente außerhalb des Mikrocomputers Platz haben.

Das PET-Programm

Das in *Bild 5* abgedruckte Programm entspricht in der Adressenbelegung der I/O-Ports dem User-Port PA des Mikrocomputers PET-2001. Da es in Basic geschrieben ist, läßt es sich ohne Schwierigkeiten auf beliebige andere Computer übertragen, wenn man nur diese Adressen entsprechend ändert.

Selbstverständlich wäre es auch möglich gewesen, die Erzeugung von großen Ziffern durch systematisches Optimieren speicherplatzsparender auszuführen. Im Interesse der Übersichtlichkeit des Programmablaufes wurde darauf aber verzichtet.

Das Programm verlangt beim Start vom Benutzer, den Spannungsbereich zu wählen. Es sei darauf hingewiesen, daß diese Abfrage ausschließlich das Anzeigeformat betrifft, nicht aber den (durch die Hardware-Beschaltung vorgegebenen) maximalen Eingangsspannungsbereich des A/D-Wandlers, der immer ±2 V beträgt und durch entsprechende Vorteilung mit Widerständen bei Bedarf verändert werden muß. Auch erlaubt die Schaltung ausschließlich die Messung von Gleichspannungen.

Willi Gietmann, Willi Billen

```
100 REM
120 REM--DIGITALVOLTMETER
130 REM
140 REM--MIT DEM 14433
170 REM--COPYRIGHT 04.NOV.1979
190 REM--WILLI GIETMANN & WILLI BILLEN
200 REM
320 PRINT"sQQBITTE WAEHLEN SIE DEN SPANNUNGSBEREICH !"
330 DIMW1(60)
340 PRINT"QQQ      2 V ..... R1"
350 PRINT"Q     20 V ..... R2"
360 PRINT"Q    200 V ..... R3"
370 PRINT"Q   2000 V ..... R4"
380 GETT$:A9=VAL(T$):IFA9<1ORA9>4THEN380
390 A9=A9*8+6
400 PRINT"sQQWIE LANGE SOLL GEMESSEN WERDEN ?"
410 PRINT"QQQQ    1 MIN ..... R1"
420 PRINT"Q    1 STD ..... R2"
430 PRINT"Q    5 STD ..... R3"
440 PRINT"Q   15 STD ..... R4"
450 PRINT"Q   24 STD ..... R5"
460 GETT$:B9=VAL(T$):IFB9<1ORB9>5THEN460
470 IFB9=1THENB9=60
480 IFB9=2THENB9=3600
490 IFB9=3THENB9=18000
500 IFB9=4THENB9=54000
510 IFB9=5THENB9=86400
520 PRINT"sQQWIE GROSS IST DIE MESSRATE ?"
530 PRINT"QQQQ    1 SEK ..... R1"
540 PRINT"Q    1 MIN ..... R2"
550 PRINT"Q    5 MIN ..... R3"
560 PRINT"Q   30 MIN ..... R4"
570 PRINT"Q   60 MIN ..... R5"
580 GETT$:C9=VAL(T$):IFC9<1ORC9>5THEN580
590 IFC9=2THENC9=60
600 IFC9=3THENC9=300
610 IFC9=4THENC9=1800
620 IFC9=5THENC9=3600
630 D9=B9/C9:PRINT"s"
640 IFD9<70THEN700
650 PRINT"QQQMESSRATE ZU HOCH UND/ODERQQ"
660 PRINT"MESSDAUER ZU NIEDRIG !QQQQQ"
670 PRINT"BITTE ANDERE WERTE WAEHLEN !"
680 FORI=1TO2000:NEXTI
690 GOTO400
700 IFD9>2THEN740
710 PRINT"QQQMESSRATE ZU NIEDRIG UND/ODERQQ"
720 PRINT"MESSDAUER ZU HOCH !QQQQQ"
730 GOTO670
740 PRINT"s"
750 FORI=826TO923:READA:POKEI,A:NEXTI
760 FORI=1000TO1015:READA:POKEI,A:NEXT
770 DATA 165,37,133,39,70,39,165,38,133,40,70,40,160,0,132,35,169
780 DATA 128,133,36,56,169,24,229,40,170,24,165,35,101,39,133,35,202
790 DATA 48,13,24,165,35,105,40,133,35,144,244,230,36,208,240,177,35
800 DATA 162,15,221,232,3,240,3,202,16,248,138,70,37,144,12,70,38
810 DATA 144,4,9,4,208,14,9,8,208,10,70,38,144,4,9,2,208
820 DATA 2,9,1,170,189,232,3,145,35,141,231,3,96
830 DATA32,123,126,97,124,255,226,236,108,98,127,252,225,254,251,160
840 PRINT"sQQQQQQQQQQQQQQQQQQQQSPANNUNGSBEREICH  ";
850 IFA9=14THENPRINT"   2 VOLT"
860 IFA9=22THENPRINT"  20 VOLT"
870 IFA9=30THENPRINT" 200 VOLT"
880 IFA9=38THENPRINT"2000 VOLT"
890 PRINT"QMESSZEIT            ";
900 IFB9=60THENPRINT" 1 MIN"
910 IFB9=3600THENPRINT" 1 STD"
920 IFB9=18000THENPRINT" 5 STD"
930 IFB9=54000THENPRINT"15 STD"
940 IFB9=86400THENPRINT"24 STD"
950 PRINT"QMESSRATE            ";
960 IFC9=1THENPRINT" 1 SEC"
970 IFC9=60THENPRINT" 1 MIN"
980 IFC9=300THENPRINT" 5 MIN"
990 IFC9=1800THENPRINT"30 MIN"
1000 IFC9=3600THENPRINT"60 MIN"
1010 POKE59459,0
1020 GOTO2490
1030 FORI=0TO3
1040 FORI=0TO3
1050 A=59471
1060 B=PEEK(A)
1070 C=BAND15
1080 IFC<>2↑ITHEN1060
1090 W(I)=INT((BAND240)/16)
1100 NEXTI
1110 Z7=TI
1120 A$=""
1130 FORI=2TO0STEP-1
1140 A$=A$+RIGHT$(STR$(W(I)),1)
1150 NEXTI
1160 IFW(3)=15ORW(3)=14THENA$="+ "+A$
1170 IFW(3)=4THENA$="+1"+A$
1180 IFW(3)=7THENA$="+1999"
1190 IFW(3)=11ORW(3)=10THENA$="- "+A$
1200 IFW(3)=0THENA$="-1"+A$
1210 IFW(3)=3THENA$="-1999"
1220 A=0
1230 PRINT"SQQQQQ"
1240 IFLEFT$(A$,1)="+"THENGOSUB2180:GOTO1260
1250 GOSUB2260
1260 FORI=2TO5
1270 PRINT"SQQQQQ"
1280 R=VAL(MID$(A$,I,1))+1
1290 A=8*I-8
1300 ONRGOSUB1380,1460,1540,1620,1700,1780,
     1860,1940,2020,2100
1310 NEXTI
1320 PRINT"qq":PRINTTAB(A9);"R"
1330 RETURN
1340 IFA9=38THEN1000
1350 PRINT"qq"
1360 PRINTTAB(A9);"R"
1370 RETURN
1380 PRINTTAB(A);"■■■■■"
1390 PRINTTAB(A);"■   ■"
1400 PRINTTAB(A);"■   ■"
1410 PRINTTAB(A);"■   ■"
1420 PRINTTAB(A);"■   ■"
1430 PRINTTAB(A);"■   ■"
1440 PRINTTAB(A);"■■■■■"
1450 RETURN
1460 PRINTTAB(A);" ■■  "
1470 PRINTTAB(A);"  ■  "
1480 PRINTTAB(A);"  ■  "
```

```
1490 PRINTTAB(A);"  ▒  "
1500 PRINTTAB(A);"  ▒  "
1510 PRINTTAB(A);"  ▒  "
1520 PRINTTAB(A);" ▒▒▒ "
1530 RETURN
1540 PRINTTAB(A);"▒▒▒▒▒"
1550 PRINTTAB(A);"▒   ▒"
1560 PRINTTAB(A);"    ▒"
1570 PRINTTAB(A);"▒▒▒▒▒"
1580 PRINTTAB(A);"▒    "
1590 PRINTTAB(A);"▒    "
1600 PRINTTAB(A);"▒▒▒▒▒"
1610 RETURN
1620 PRINTTAB(A);"▒▒▒▒▒"
1630 PRINTTAB(A);"    ▒"
1640 PRINTTAB(A);"    ▒"
1650 PRINTTAB(A);"  ▒▒▒"
1660 PRINTTAB(A);"    ▒"
1670 PRINTTAB(A);"    ▒"
1680 PRINTTAB(A);"▒▒▒▒▒"
1690 RETURN
1700 PRINTTAB(A);"▒  ▒  "
1710 PRINTTAB(A);"▒  ▒  "
1720 PRINTTAB(A);"▒  ▒  "
1730 PRINTTAB(A);"▒▒▒▒▒"
1740 PRINTTAB(A);"    ▒  "
1750 PRINTTAB(A);"    ▒  "
1760 PRINTTAB(A);"    ▒  "
1770 RETURN
1780 PRINTTAB(A);"▒▒▒▒▒"
1790 PRINTTAB(A);"▒    "
1800 PRINTTAB(A);"▒▒▒▒▒"
1810 PRINTTAB(A);"    ▒"
1820 PRINTTAB(A);"    ▒"
1830 PRINTTAB(A);"    ▒"
1840 PRINTTAB(A);"▒▒▒▒▒"
1850 RETURN
1860 PRINTTAB(A);"▒▒▒▒▒"
1870 PRINTTAB(A);"▒   ▒"
1880 PRINTTAB(A);"▒    "
1890 PRINTTAB(A);"▒▒▒▒▒"
1900 PRINTTAB(A);"▒   ▒"
1910 PRINTTAB(A);"▒   ▒"
1920 PRINTTAB(A);"▒▒▒▒▒"
1930 RETURN
1940 PRINTTAB(A);"▒▒▒▒▒"
1950 PRINTTAB(A);"    ▒"
1960 PRINTTAB(A);"    ▒"
1970 PRINTTAB(A);"    ▒"
1980 PRINTTAB(A);"    ▒"
1990 PRINTTAB(A);"    ▒"
2000 PRINTTAB(A);"    ▒"
2010 RETURN
2020 PRINTTAB(A);"▒▒▒▒▒"
2030 PRINTTAB(A);"▒   ▒"
2040 PRINTTAB(A);"▒   ▒"
2050 PRINTTAB(A);"▒▒▒▒▒"
2060 PRINTTAB(A);"▒   ▒"
2070 PRINTTAB(A);"▒   ▒"
2080 PRINTTAB(A);"▒▒▒▒▒"
2090 RETURN
2100 PRINTTAB(A);"▒▒▒▒▒"
2110 PRINTTAB(A);"▒   ▒"
2120 PRINTTAB(A);"▒   ▒"
2130 PRINTTAB(A);"▒▒▒▒▒"
2140 PRINTTAB(A);"    ▒"
2150 PRINTTAB(A);"    ▒"
2160 PRINTTAB(A);"▒▒▒▒▒"
2170 RETURN
2180 PRINTTAB(A);"      "
2190 PRINTTAB(A);"  ▒   "
2200 PRINTTAB(A);"  ▒   "
2210 PRINTTAB(A);"▒▒▒▒▒"
2220 PRINTTAB(A);"  ▒   "
2230 PRINTTAB(A);"  ▒   "
2240 PRINTTAB(A);"      "
2250 RETURN
2260 PRINTTAB(A);"      "
2270 PRINTTAB(A);"      "
2280 PRINTTAB(A);"      "
2290 PRINTTAB(A);"▒▒▒▒▒"
2300 PRINTTAB(A);"      "
2310 PRINTTAB(A);"      "
2320 PRINTTAB(A);"      "
2330 RETURN
```

```
2340 PRINT"♥[% "
2350 PRINT"Q100"
2360 PRINT"·QQQQQQQQQ50"
2370 PRINT"  QQQQQQQQQ0"
2380 PRINT"S    ";
2390 FORI=1TO24:PRINT"|Q|";:NEXTI
2400 PRINT"SQQ   ";
2410 PRINT"┤"
2420 PRINT"QQQQQQQQQ   ";
2430 PRINT"┤"
2440 PRINT"QQQQQQQQQ   ";
2450 PRINT"+";
2460 FORI=1TO36:PRINT"—";:NEXTI
2470 PRINT"S"
2480 GOTO2580
2490 FORI9=1TOD9
2500 Z=TI
2510 IF(TI-Z)/60<C9THENGOSUB1030:GOTO2510
2520 FORJ1=2TO5
2530 H=VAL(MID$(A$,J1,1))
2540 W1(I9)=W1(I9)+H*10↑(5-J1)
2550 NEXTJ1
2560 NEXTI9
2570 GOTO2340
2580 FORI=1TOD9
2590 X=I+7
2600 Y=W1(I)/50+5
2610 IFY<6THENY=6
2620 POKE37,X:POKE38,Y
2630 SYS826
2640 NEXTI
2650 GOTO2650
2660 DIMW1(60)
2670 FORI=1TO20:W1(I)=0:NEXTI
2680 FORI=21TO40:W1(I)=1000:NEXTI
2690 FORI=41TO60:W1(I)=2000:NEXTI
2700 D9=60:GOTO2570
READY.
```

Bild 5. Das Programm erläutert sich weitgehend selbst. Zu beachten ist, daß die Abfrageadressen für den User-Port und andere PEEKs und POKEs natürlich nur für den PET-2001 gelten

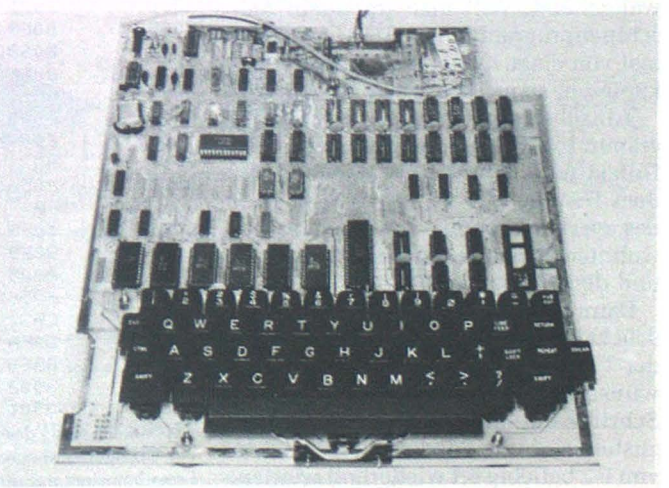
Preiswert um jeden Preis

Das „Superboard" von Ohio Scientific

Die derzeit mit Abstand preiswerteste Möglichkeit, einen in Basic programmierbaren Computer zu betreiben, ist das „Superboard" alias Challenger 1 P – letzterer ist praktisch ein Superboard mit Gehäuse und Netzteil. Über eine Videobuchse oder einen UHF-Modulator kann man ein handelsübliches Fernsehgerät anschließen, dessen Bildwechselfrequenz allerdings mit dem Vertikalregler auf 60 Hz umgestellt werden muß. Auf dem Bildschirm werden – und das ist schon ein erster Mangel – 24 Zeilen mit nur 24 Zeichen pro Zeile dargestellt; die Zeichen sehen dadurch sehr „plattgedrückt" aus. Bei Verwendung einigermaßen guter 31-cm-Monitoren ist sogar die horizontale Punktstruktur der einzelnen Zeichen deutlich zu erkennen, so daß sich eher ein Mini-Fernsehgerät mit 11...13 cm Bilddiagonale empfiehlt. Bereits geringe Brummanteile auf den Versorgungsspannungen von Computerplatine oder Fernsehgerät führen zu einem äußerst unangenehmen Flimmern mit 10 Hz (Differenz von 50 Hz und 60 Hz).

Hat man aber einmal das Superboard in Betrieb genommen – dazu genügt eine einzige Versorgungsspannung von 5 V (3 A) –, kommt man in den Genuß eines recht komfortablen Microsoft-Basic, das demjenigen von PET, CBM und AIM nahekommt. Der Interpreter belegt 8 KByte ROM und ermöglicht auch Stringoperationen wie bei den vorgenannten Geräten. Etwas merkwürdig ist das Fehlen eines GET-Befehls, um einzelne Zeichen von der Tastatur einlesen zu können. An Winkelfunktionen sind ATN, COS und SIN vorhanden; alle anderen können jedoch aus diesen abgeleitet werden. Der Rechenbereich beträgt $10^{-32}...10^{+32}$.

Ungewohnt ist auch die Benutzung von LOAD und SAVE. Im Gegensatz zu den meisten anderen Computern kann das „Superboard" sich nicht selbst aus mehreren Programmen einer Kassette



Das „Superboard" ist ein mit einem Microsoft-Basic-Interpreter ausgerüsteter Einplatinen-Computer mit Video-Ausgang

das richtige heraussuchen, weil diese nicht mit Namen gekennzeichnet werden können. LOAD schaltet schlicht und einfach die Basic-Eingabe vom Tastenfeld auf das Kassetteninterface um, und SAVE sorgt dafür, daß die Ausgabe aller Informationen zusätzlich zum Bildschirm auch auf Kassette erfolgt, und zwar mit 300 bit/s im Kansas-City-Standard; das entspricht 30 Byte/s.

Die Schaltung für das Kassetten-Interface arbeitet mit einem UART/ACIA 6850 zur Parallel-/Serien-Wandlung, mehreren TTL-Bausteinen sowie einem Operationsverstärker. Dies ist ein wenig verwunderlich, weil es ja kein Problem gewesen wäre, die für den Kansas-City-Standard erforderlichen Töne 1200 Hz (0) und 2400 Hz (1) per Software zu erzeugen und auszuwerten. Auf der Platine befindet sich schließlich noch ein „Expansion Connector" für den Anschluß von Speichererweiterungen oder einer Floppy-Disk. Er hat die Form einer 40poligen IC-Fassung.

Daß der Computer eine Syntax-Fehlermeldung nach dem Laden von Programmen ausgibt, obwohl gar kein Fehler vorliegt, ist eine weitere Eigenheit. Die Fehlermeldungen werden allerdings nicht wie etwa beim PET in voller Länge oder wie beim AIM als sinnvolle Abkürzung ausgegeben, sondern als Folge eines Buchstabens und eines Grafikzeichens, was das Verständnis nicht unbedingt erleichtert.

Das Superboard ist primär für das Arbeiten in Basic konstruiert. So ist denn auch der Maschinensprache-Monitor recht unkomfortabel ausgefallen: Der Benutzer kann Speicherzellen ansehen und ändern sowie ein Maschinensprache-Programm starten. Leider ist die Dokumentation auch der ROM-Software aber so mangelhaft, daß man davon zunächst ohnehin die Finger lassen sollte. Verwendet wird übrigens die CPU 6502 mit 4-KByte-RAM in der Grundausstattung, wobei eine Erweiterung auf der Platine bis zu 8-KByte-RAM möglich ist.

Dem Superboard merkt man also recht deutlich an, daß es speziell auf Preiswürdigkeit gezüchtet wurde und zudem ein recht frühes Produkt des Hobbycomputer-Marktes ist. Immerhin stellt diese Platine aber derzeit die preisgünstigste Möglichkeit dar, erste Schritte in der Programmiersprache Basic zu wagen. Fe.

AIM kopiert Kassetten

Das Kopieren von Maschinenprogrammen, die auf Kassette abgespeichert sind, ist bei den (praktisch identischen) Computern AIM-65 und PC-100 meist recht langwierig – besonders dann, wenn man die angesprochenen Adressenbereiche nicht kennt. Zwar stellt das „AIM-Directory" aus dem FUNKSCHAU-Sonderheft „Programme" eine deutliche Erleichterung dar, indem es diese Bereiche ausdruckt; trotzdem bleibt noch das umständliche Eingeben der Adressen von Hand.

Das kleine Programm im *Bild* enthebt den AIM-Benutzer von solcher Arbeit weitgehend. Will man irgendein Maschinenprogramm im AIM-Objektformat von einer Kassette auf eine andere kopieren, so geht man wie folgt vor:

Zunächst muß man das Kopierprogramm lt. Listing eingeben oder laden. Drückt man dann F1, so erscheint auf dem Display „IN =". Nun kann man das zu kopierende Programm in gewohnter Weise laden, indem man T und den Filenamen eingibt.

Dann tauscht man die Kassetten aus, drückt F 2 und gibt nach „OUT =" wieder T und den für das Abspeichern gewünschten Filenamen ein. Dieser Schritt kann, um mehrere Kopien herzustellen, durch wiederholtes Drücken von F 2 beliebig oft wiederholt werden.

Die Adressenbelegung des Programms (0000...006C) überschneidet sich nicht mit dem Menü-Programm, das ebenfalls in dem erwähnten Sonderheft „Programme" veröffentlicht ist und zum schnellen Auffinden bestimmter Bandstellen dient. Das eigentliche Programm beginnt bei 0004; man darf nicht vergessen, auch die F1/F2-Vektoren bei 010C einzugeben. Die Zellen 0000...0003 dienen als Adressenzähler sowie zum Zwischenspeichern der Endadresse für den Pufferspeicher, der hier bei 0200 beginnt.

Das zu kopierende Programm wird nicht in seine ursprünglichen Adressenbereiche geladen und kann daher auch nicht gestartet werden. Vielmehr liegt es im Pufferspeicher ab 0200 inklusive aller Steuerzeichen (Blockadressen, Byteanzahl usw.). Deshalb kann man auch Programme kopieren, die in der vom Kopierprogramm belegten Page Zero oder in Bereichen liegen, für die im Benutzersystem kein RAM vorhanden ist. Bei der 4-KByte-AIM-Version können allerdings nur Programme bis zu etwa 3 KByte Länge kopiert werden. Damit der Verify-Befehl (Taste 3) funktioniert, muß am Programmende der RTS-Befehl geändert werden auf: 006C 4C F0 E9.

In der vorliegenden Version eignet sich das Kopierprogramm nur für das AIM-Objektformat, nicht aber für das KIM-, Editor- oder Basic-Format. Allerdings ist bei letzteren das Kopieren ohnehin problemlos, weil nicht unterschiedliche Adressenbereiche eingegeben werden brauchen. Fe

```
X          PASS 1
0112       PASS 2
0000 DU12          =$E511
0000 WHI           =$E848
0000 WHO           =$E871
0000 IN            =$E993
0000 OUT           =$E9BC
0000 BUF           =$200
0000               *=0
0000 PNT           *=*+2
0002 END           *=*+2
0004 LOAD  206400  JSR SETP
0007       2048E8  JSR WHI
000A L1    2093E9  JSR IN
000D       C93B    CMP #';
000F       D0F9    BNE L1
0011       205900  JSR STP+3
0014 L2    205600  JSR STP
0017       AA      TAX
0018       F00B    BEQ S1
001A       6903    ADC #3
001C       AA      TAX
001D L3    205600  JSR STP
0020       CA      DEX
0021       10FA    BPL L3
0023       30E5    BMI L1
0025 S1    A205    LDX #5
0027       205600  JSR STP
002A       CA      DEX
002B       D0FA    BNE S1+2
002D       A500    LDA PNT
002F       8502    STA END
0031       A501    LDA PNT+1
0033       8503    STA END+1
0035 M     4C11E5  JMP DU12
0038 SAVE  2071E8  JSR WHO
003B       206400  JSR SETP
003E S2    A000    LDY #0
0040       B100    LDA (PNT),Y
0042       20BCE9  JSR OUT
0045       205D00  JSR INCP
0048       A500    LDA PNT
004A       C502    CMP END
004C       D0F0    BNE S2
004E       A501    LDA PNT+1
0050       C503    CMP END+1
0052       D0EA    BNE S2
0054       F0DF    BEQ M
0056 STP   2093E9  JSR IN
0059       A000    LDY #0
005B       9100    STA (PNT),Y
005D INCP  E600    INC PNT
005F       D002    BNE *+4
0061       E601    INC PNT+1
0063       60      RTS
0064 SETP  A900    LDA #<BUF
0066       8500    STA PNT
0068       A902    LDA #>BUF
006A       8501    STA PNT+1
006C       60      RTS
006D               *=$10C
010C       4C0400  JMP LOAD
010F       4C3800  JMP SAVE
0112               .END
0112         ERRORS= 0000
```

Assembler-Listing des Kopierprogramms für AIM-65 und PC-100. Es gestattet das problemlose Kopieren von Objektprogrammen selbst dann, wenn man die angesprochenen Adressenbereiche nicht kennt. Eilige Leser brauchen nur die Bytes in der mittleren Spalte ab 0004 einzugeben (20 64 00 20 usw.)

Ursula und Lutz May

16-KByte-Speicherkarte

1. Teil

Wenn ein Mikroprozessorsystem mit höheren Programmiersprachen wie Basic, Assembler usw. arbeiten soll, so stellt man bei kleineren Anlagen bald fest, daß der zur Verfügung stehende Arbeitsspeicher nicht ausreicht. Der nachfolgende Beitrag beschreibt den Zusammenbau eines bis 16 KByte erweiterbaren statischen Datenspeichers, der sich für 8-bit-Mikroprozessorsysteme eignet und im Zusammenhang mit dem in [1] beschriebenen 6802-Computer entwickelt wurde.

Produktüberlegungen

Im allgemeinen haben Anwenderprogramme, die in Maschinensprache geschrieben wurden, eine durchschnittliche Länge von 1 bis 2 KByte. Der dazu notwendige Arbeitsspeicher ist selbst in einfacheren Computersystemen vorhanden. Soll jedoch das gleiche Programm über einen Assembler geschrieben werden, dann steigt der Speicherbedarf auf 16...32 KByte an. Hat der Software-Entwickler dabei den Wunsch, sein in Assembler geschriebenes Programm umfangreich zu dokumentieren, dann erhöhen sich die hier angegebenen Werte. Bei Systemen, die mit Basic arbeiten, sieht es ähnlich aus: Sobald eine größere Datenmenge zu verarbeiten ist, wie es z. B. bei einem Lagerverwaltungsprogramm oder einer Adressenkartei der Fall wäre, muß mit einem hohen Speicherbedarf gerechnet werden.

Aus dieser Notwendigkeit heraus wurde eine Speicherkarte im Europaformat entwickelt, die in 1-KByte-Schritten zu einer 16-KByte-RAM-Baugruppe aufgerüstet werden kann und für 8-bit-Mikroprozessorsysteme mit einem 16-bit-Adreßbus verwendbar ist.

Bei der hier beschriebenen Baugruppe wurden statische Speicherelemente verwendet, um die Kosten für einen kleineren Speicher niedrig halten zu können. Die Bus-Steckerbelegung ist Pin-gleich mit der der 6802-Computerkarte [1]. Als Verbindungselement dient eine dreireihige, 64polige VG-Steckerleiste, bei der nur die äußeren Stiftreihen bestückt sind. Aus der *Tabelle 1* ist die genaue Belegung zu ersehen. *Tabelle 2* gibt Auskunft über allgemeine technische Daten.

Schaltungsbeschreibung

Die Schaltung besteht aus drei Baugruppen: Adreßdecoder, Daten- und Adreßbuspuffer und dem Speicher.

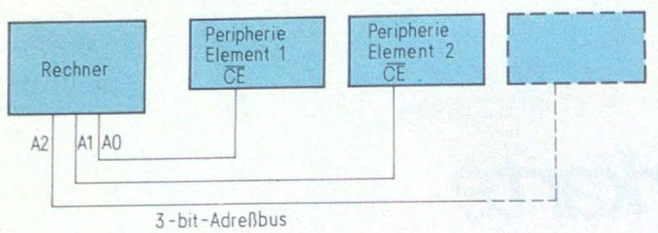
Der Adreßdecoder

Bei einem Mikrocomputer-Minimalsystem [1], bestehend aus einem Mikroprozessor, Daten- und Programmspeicher und einer Peripherieschnittstelle, werden zur genauen Adressendefinition der einzelnen Baugruppen, die vom Rechnerbaustein angesprochen werden sollen, keine weiteren Decodereelemente benötigt. Der 16-bit-Adreßbus reicht zur einwandfreien Selektierung der wenigen Elemente völlig aus. Kommen jedoch weitere Baugruppen hinzu, so müssen mit einem entsprechenden Hardware-Aufwand die für die einzelnen Bauelemente vom Konstrukteur zugeordneten Adressen genau decodiert werden können.

In einem Beispiel soll das verdeutlicht werden: Mit einem aus drei Leitungen bestehenden Adreßbus sollen zwei verschiedene Baugruppen eindeutig adressiert werden können. Der hier frei erfundene 3-bit-Adreßbus ermöglicht die Bestimmung von acht verschiedenen Adressen (*Tabelle 3*). Eine

Tabelle 1: Steckerbelegung

Pin	Verwendung	
1c	+ 5 V	Versorgungsspannung
1a	+ 5 V	Versorgungsspannung
7c	A0	niederwertigstes Adressenbit
7a	A1	16-bit-Adreßbus
8c	A2	↓
8a	A3	↓
9c	A4	↓
9a	A5	↓
10c	A6	↓
10a	A7	↓
11c	A8	↓
11a	A9	↓
13c	A10	↓
13a	A11	↓
14c	A12	↓
14a	A13	↓
15c	A14	↓
15a	A15	höchstwertiges Adressenbit
22a	VMA	gültige Adresse
23c	R/$\overline{W}$	Lese-Schreib-Steuerleitung
23a	02	Rechnerarbeitstakt
24c	D0	niederwertigstes Datenbit
24a	D1	8-bit-Datenbus
25c	D2	↓
25a	D3	↓
26c	D4	↓
26a	D5	↓
27c	D6	↓
27a	D7	höchstwertiges Datenbit
32c	0 V	Masse (Versorgungsspannung)
32a	0 V	Masse (Versorgungsspannung)



▲ Bild 1. Einfachste Selektierung von Mikrocomputer-Bauelementen am Beispiel eines 3-bit-Adreßbus. CE = Chip Enable = Freigabe des Bauelements beim Anliegen einen L-Signals (0 V)

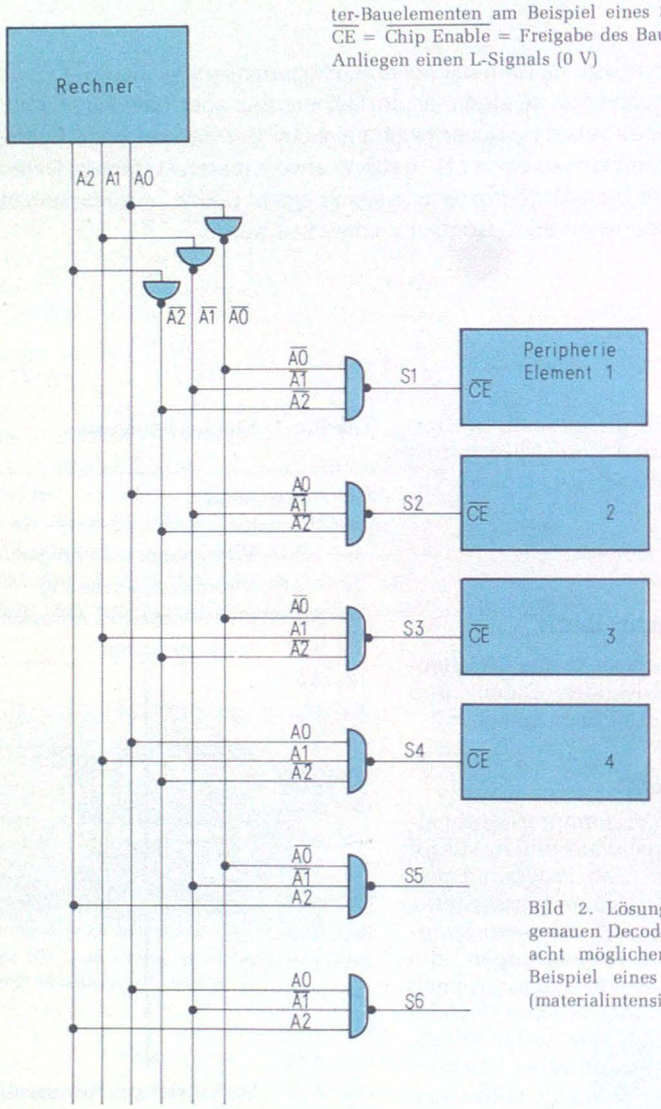


Bild 2. Lösungsvorschlag zur genauen Decodierung einer der acht möglichen Adressen am Beispiel eines 3-bit-Adreßbus (materialintensive Schaltung)

Baugruppe wird durch das Anlegen eines Null-Potentials angesprochen.

Durch Verbinden der zwei Peripheriebaugruppen mit jeweils einer der drei möglichen Adreßleitungen ist sichergestellt, daß der Rechnerbaustein jedes Element einzeln ansteuern kann (*Bild 1*).

Es besteht auf diese Weise die Möglichkeit, noch eine dritte Einheit anzuschließen. Bei der Programmierung des Rechners dürfte bei dieser Problemlösung nur ein Teil der acht möglichen Adressen ausgegeben werden, da hier eine Fehladressierung möglich ist. Es dürfen nur die Codes der Adressen 4, 6 und 7 verwendet werden (Tabelle 3). Alle anderen Codes sprechen zwei oder sogar drei Elemente auf einmal an.

Soll der Rechner nun noch ein viertes oder fünftes Element ansteuern, dann ist eine genauere Adressendecodierung notwendig, da das erstgenannte Verfahren keine weiteren Möglichkeiten bietet. In *Bild 2* wird eine Decodiermöglichkeit aufgezeigt, die sehr materialintensiv ist, aber dafür recht überschaubar. Mit drei Inverterelementen werden die drei Adressenleitungen negiert, so daß dem Anwender zur Bestimmung einer Adresse sechs verschiedene Leitungen zur Verfügung stehen. Mit Dreifach-NAND-Gattern kann er nun jede der acht möglichen Adressen selektieren und damit die Peripherieelemente ansteuern (*Tabelle 4*). Eine schaltungstechnisch einfachere Lösung stellt *Bild 3* dar. Hier wird der Decoder-Baustein 74LS138 eingesetzt.

Tabelle 2: Technische Daten der 16-K-RAM-Karte

Kartenformat	100 mm x 160 mm (Europa)
Busstecker	64polige VG-Steckerleiste; Reihe a+c belegt
Versorgungsspannung	+ 5 V
Stromaufnahme	ohne Speicherbausteine 260 mA (nur Puffer und Adreßdecoder) je 1 KByte (2 x 2114 stand.): + 160 mA je 1 KByte (2 x 2114 L): + 120 mA je 1 KByte (2 x Harris MI-6514): + 14 mA
Speicherkapazität	in 1-KByte-Schritten von 1 KByte bis 16 KByte erweiterbar
Datenzugriffszeit	abhängig vom verwendeten Speichertyp 450...200 ns

Tabelle 3: Wahrheitstabelle des 3-bit-Adreßbus

A2	A1	A0	Adresse
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

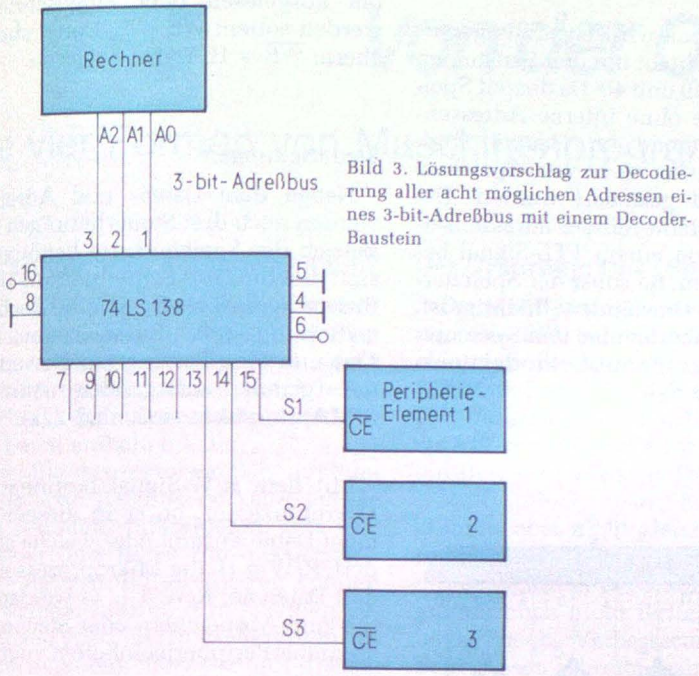


Bild 3. Lösungsvorschlag zur Decodierung aller acht möglichen Adressen eines 3-bit-Adreßbus mit einem Decoder-Baustein

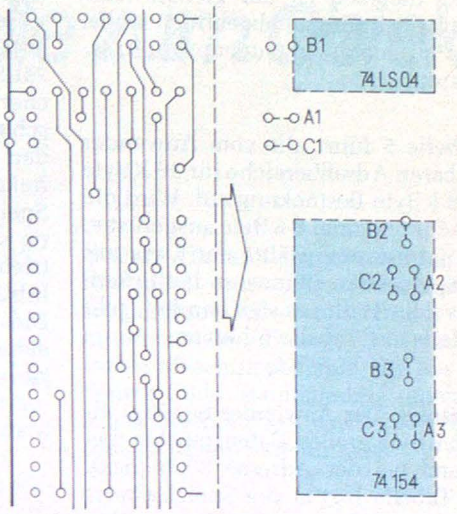


Bild 4. Linke Grafik: Ein Leiterbahnenausschnitt im vergrößerten Maßstab von der Lötseite betrachtet, daneben der gleiche Ausschnitt, wobei nur noch die zur Brückenbildung notwendigen Lötaugenpaare dargestellt sind. In Pfeilrichtung befindet sich die 64polige DIN-Steckerleiste

Tabelle 4: Adressendecodierung mit drei Invertern und NAND-Gattern

A2	A1	A0	A2	A1	A0	Adr.	Realisierung
0	0	0	1	1	1	1	$S_1 = \overline{A_0} \cdot \overline{A_1} \cdot \overline{A_2}$
0	0	1	1	1	0	2	$S_2 = A_0 \cdot \overline{A_1} \cdot \overline{A_2}$
0	1	0	1	0	1	3	$S_3 = \overline{A_0} \cdot A_1 \cdot \overline{A_2}$
0	1	1	1	0	0	4	$S_4 = A_0 \cdot A_1 \cdot \overline{A_2}$
1	0	0	0	1	1	5	$S_5 = \overline{A_0} \cdot \overline{A_1} \cdot A_2$
1	0	1	0	1	0	6	$S_6 = A_0 \cdot \overline{A_1} \cdot A_2$
1	1	0	0	0	1	7	$S_7 = \overline{A_0} \cdot A_1 \cdot A_2$
1	1	1	0	0	0	8	$S_8 = A_0 \cdot A_1 \cdot A_2$

Ähnlich wie in den beiden letzten Beispielen werden auch die Steueradressen der 16-KByte-Speicherkarte decodiert. Drei Inverter und ein Achtfach-NAND-Gatter (IC 36) bestimmen den Adressenbereich der Speicherkarte. Der Anwender hat hierbei die Möglichkeit, mit drei Drahtbrücken diesen Bereich selbst zu definieren. Dabei kann er auch entscheiden, ob der gesamte Umfang von 16 KByte oder nur 8 KByte des Speichers belegt werden sollen.

Es müssen dazu von der Lötseite in vorbereitete Lötaugenpaare kleine Drahtbügel eingesetzt und verlötet werden (unterhalb der Bausteine 74154 [IC35] und 74LS04 [IC33]). Soll später einmal der Adressierungsbereich verlegt werden, kann man die Brückenpositionen ändern. *Bild* 4 zeigt den Leiterbahnenausschnitt unterhalb des Deco-

Tabelle 5: Speicherkarten Adreßbereichsdefinition

Adreßbereich	maximale Speichergröße	zu verbindende Lötaugenpaare									zu bestückende Speicherhälfte		
		A1	A2	A3	B1	B2	B3	C1	C2	C3	rechte	linke	beide
0000...3FFF	16 KByte	x				x			x				x
4000...7FFF	16 KByte	x			x		x		x				x
8000...BFFF	16 KByte	x				x		x		x			x
C000...FFFF	16 KByte	x			x		x	x		x			x
0000...1FFF	8 KByte		x			x			x		x		
2000...3FFF	8 KByte	x		x		x			x			x	
4000...5FFF	8 KByte		x		x		x		x		x		
6000...7FFF	8 KByte	x		x	x		x		x			x	
8000...9FFF	8 KByte		x			x		x		x	x		
A000...BFFF	8 KByte	x		x		x		x		x		x	
C000...DFFF	8 KByte		x		x		x	x		x	x		
E000...FFFF	8 KByte	x		x	x		x	x		x		x	

x = das in dieser Spalte genannte Lötaugenpaar muß mit einem kleinen Drahtbügel verbunden werden.

derbausteins IC 35 und Inverters IC 33. Rechts daneben ist der gleiche Ausschnitt noch einmal abgebildet, wobei hier nur noch die Lötaugenpaare zu sehen sind.

Tabelle 5 führt alle vom Anwender wählbaren Adreßbereiche für 16 KByte und 8 KByte Bestückung auf. Wird die Karte für maximal 8 KByte ausgerüstet, dann ist nur eine Hälfte der Karte mit den Speicherbausteinen zu bestücken. Um welche Hälfte es sich handelt, geht ebenfalls aus Tabelle 5 hervor.

Beispiel: Der Anwender benötigt einen 6 KByte großen Datenspeicher, beginnend bei der Adresse 2000 (hex). Laut Tabelle legt er die Speicherkarte für einen 8 KByte großen Adressenbereich aus und verbindet folgende Lötaugenpaare mit kleinen Drahtbrücken: A1, A3, B2 und C2. Damit belegt diese Baugruppe den Adressenbereich von 2000...3FFF (hex), der nun von keinem anderen System mehr mitbeansprucht werden darf.

Vom Busstecker aus betrachtet, wird nur die linke Kartenhälfte in aufsteigender Folge mit 16 Speicher-ICs (2114) bestückt. Die Plazierung der mit A1...A3, B... und C... bezeichneten Lötaugenpaare wird in *Bild 5* dargestellt.

Innerhalb des ausgewählten Adressenbereichs der Speicherkarte müssen die Steueradressen in 1-K-Schritten zur Selektierung der einzelnen 1-K-Speicherbausteine gebildet werden. Dazu wird der Decoder 74154 (IC 35) verwendet, der mit den Adreßleitungen A10...A13 vom Rechnerbus angesteuert wird. Der Decoder bildet daraus die 16 Speicherfreigabe-Signale $\overline{CS}$ ($\overline{\text{Chip select}}$). Diese CS-Signale können aber nur dann abgegeben werden, wenn vom Adreßbereichsdecoder über IC 36, Pin 8, das Speicherfreigabe-Signal anliegt (= L).

Daten- und Adreßbuspuffer

Das Speicherkartenfreigabe-Signal von IC 36, Pin 8, wird auch zur Ansteuerung des Datenbuspuffers, 2 x 8T26 oder 2 x MC 6880 (IC 37 und 38), benötigt. Nur wenn eine für die Karte gültige Adresse vorliegt, darf der Datenbuspuffer aktiviert werden, wobei die Datenrichtung der von ihm zu übertragenden Informationen mit der R/$\overline{W}$-Steuerleitung bestimmt wird. R/$\overline{W}$ ist die Lese-Schreib-Steuerleitung, die vom Mikroprozessor aus verwaltet wird.

Der Adreßbus wird ebenfalls gepuffert. Dies geschieht mit den Bausteinen 74LS367, IC 39 und 40. Da derzeit Speicherbausteine ohne interne Adressenschutzwiderstände im Handel sind, darf dieser Puffer nicht vom Speicherfreigabesignal gesteuert werden. Die Speicherbausteine müssen adressenseitig ständig von einem TTL-Signal betrieben werden, da sonst der Speicherinhalt unter Umständen flüchtig ist. Dies ist vornehmlich bei den Speicherelementen aus Fernost, Produktionsjahr 1978, der Fall.

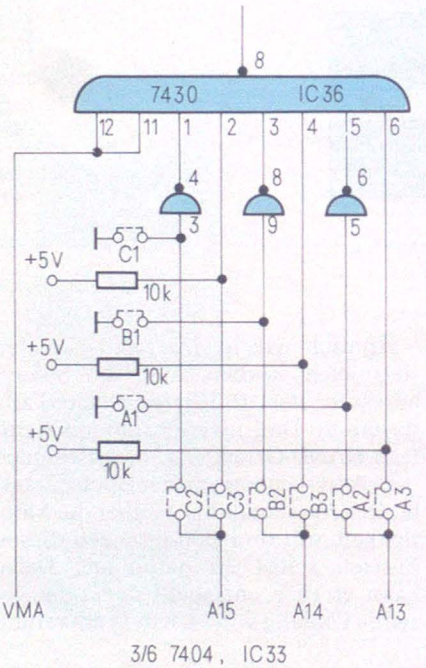


Bild 5. Adreßdecoder der Speicherkarte mit der genauen Plazierung der einzelnen Lötaugenpaare

Der Speicher

Zur Speicherung der Daten wird der inzwischen sehr preisgünstige Baustein 2114 verwendet, der 1024 x 4 bit Speicherplätze enthält. Zur Bildung eines 1 K x 8-bit-Speichers werden daher immer zwei Elemente benötigt. Der Speicherbaustein wird von mehreren Halbleiterherstellern unter der Bezeichnung 2114 vertrieben, wobei es von Harris einen sehr stromsparenden Typ mit der Bezeichnung MI-6514 gibt. Die Speicherzugriffszeiten und der Stromverbrauch sind aus der Tabelle 2 zu entnehmen.

Über den Anschluß 8 ($\overline{CS}$) wird das angewählte Speicher-IC aktiviert, und über Pin 10 ($\overline{WE}$) wird bestimmt, ob Daten ausgelesen oder eingeschrieben werden sollen: $\overline{WE}$ = L: Daten abspeichern; $\overline{WE}$ = H: Daten auslesen.

Steuerleitungen

Neben dem Daten- und Adreßbus werden noch drei Steuerleitungen zum Betrieb der Speicherkarte benötigt. Es sind dies die Lese-Schreibleitung (R/$\overline{W}$, Bussteckeranschluß 23c), die Rechnertaktleitung (02, Bussteckeranschluß 23a) und eine Leitung zur Bewertung der gerade anliegenden Adressen (VMA, Bussteckeranschluß 22a).

Mit dem R/$\overline{W}$-Signal bestimmt der Mikroprozessor, ob er in diesem Moment Daten ausgibt oder welche anfordert: R/$\overline{W}$ = H: der Mikroprozessor fordert Daten an; R/$\overline{W}$ = L: es werden Daten zum Abspeichern oder Steuern bestimmter Peripherieeinheiten vom Prozessor ausgegeben.

Die Φ2-Taktleitung dient zum synchronisierten Arbeiten mit dem Rechner. Nur in der H-Phase (positives Potential) wird die Speicherkarte freigegeben und der Datenpuffer aktiviert.

Bei verschiedenen Mikroprozessorsystemen werden während der Bearbeitung eines gerade laufenden Befehls über eine Zeitdauer von einem oder mehreren Taktzyklen unaktuelle Adressen über den Adreßbus ausgegeben. Dies geschieht zum Beispiel auch bei bestimmten Indexregisteranweisungen vom 6800. Damit die angeschlossenen Peripherie-Elemente sich auf Grund einer solchen Pseudo-Adresse nicht angesprochen fühlen, wird über die VMA-Steuerleitung diesen Elementen mitgeteilt, ob es bei der auf dem Adreßbus befindlichen Information um eine gültige Adresse geht oder nicht: VMA = H: Adresse ist für die Peripherie-Elemente (z. B. Speicher) gültig; VMA = L: ungültige Adresse. Ist dieser Anschluß mit einem Low-Potential belegt, dann kann der Speicher von außen nicht angesprochen werden. Die Verwendung dieser Steuerleitung ist nicht unbedingt notwendig, sie kann daher vom Anwender auf H-Potential gelegt werden.

Wird diese Speicherkarte in Verbindung mit der in [1] beschriebenen Computerkarte M 6802 verwendet, so entstehen bei der Verdrahtung dieser Steuerleitungen keine Probleme.

(Schluß folgt)

Von der Quelle zum Objekt

Die vier Formen von Maschinenprogrammen

Was ist ein Maschinenprogramm?

Jedes Mikrocomputer-System besteht aus einem Mikroprozessor (CPU), einem Festwertspeicher mit dem Betriebsprogramm und einem Arbeitsspeicher, der Anwenderprogramme und Daten enthalten kann.

Der Mikroprozessor versteht nur relativ einfache Befehle; er kann gerade eben addieren, subtrahieren, Bits verschieben und Bytes transportieren – multiplizieren und dividieren können nur wenige, speziell fürs Rechnen gezüchtete Typen. Das Betriebssystem des Mikrocomputers muß also mit diesen, für die CPU direkt verständlichen Befehlen geschrieben sein. Das nennt man Maschinensprache bzw. Maschinenprogramm.

Die in Maschinensprache geschriebenen Betriebsprogramme von Basic-Mikrocomputern übersetzen die weit komfortableren Befehle der Programmiersprache Basic in eine Folge von Maschinenbefehlen. Ein einziger Basic-Befehl kann hundert Maschinenbefehlen entsprechen, und seine Ausführung dauert daher relativ lange.

In vielen Anwendungsfällen, wo es auf hohe Verarbeitungsgeschwindigkeit ankommt, verzichtet man deshalb auf die Vorteile höherer Programmiersprachen wie Basic und programmiert in Maschinensprache. Auch in der FUNKSCHAU sind häufig Maschinenprogramme zu finden, allerdings in recht unterschiedlicher Form.

Der Ursprung: Das „Sourcelisting"

Der Mikroprozessor versteht im Prinzip nur Bitkombinationen, die üblicherweise hexadezimal, d. h. mit den Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, dargestellt werden. Eine Ziffer entspricht vier Bits, bei 8-bit-CPUs besteht ein Byte also aus zwei Hex-Ziffern.

Nun pflegt man Maschinenprogramme aber nicht gleich als Hex-Ziffern hinzuschreiben, weil das viel zu unübersichtlich wäre, solange das Programm noch nicht fertig ist und vielleicht noch Verbesserungen bedarf. Statt dessen schreibt man ein Quellenprogramm (*Bild 1*), das die einzelnen Befehle in „mnemonischer" und Adressen in symbolischer Form enthält. Dieses Quellenprogramm (Sourcelisting, engl. Source = Quelle) kann auch Kommentare enthalten, die nur dem Verständnis des Programmierers dienen und von den Befehlen z. B. durch einen Strichpunkt getrennt sind.

Übersetzen mit dem Assembler

Das Quellenprogramm kann der Mikroprozessor leider noch nicht verstehen, weil es noch nicht die tatsächlichen Befehlscodes und Adressen enthält, sondern nur ihre Namen. Ein besonderes, in Maschinensprache geschriebenes Programm namens „Assembler" übersetzt nun das Sourcelisting in die für die CPU verständliche Maschinensprache, ins „Objektprogramm". Dabei wird gleichzeitig (zur Kontrolle) ein „Assembler-Listing" ausgegeben, das – wie in *Bild 2* – die hexadezimale Adresse, die hexadezimalen Maschinenbefehle, die aus dem Source-Code stammenden mnemonischen Befehle (LDA usw.) sowie eventuelle Kommentare enthält.

Am schnellsten einzutippen: Das Maschinenprogramm

Ist das Programm erst einmal fertig assembliert, d. h. übersetzt, so steht es im Speicher des Mikrocomputers tatsächlich nur als Folge hexadezimaler Bytes (*Bild 3*). Es ist leicht einzusehen, daß der Abdruck ein und desselben Programms als Assemblerlisting am meisten, als reines Hex-Listing aber am wenigsten Platz beansprucht. Auch muß man bei der Eingabe des Programms als Hex-Listing weit weniger Tasten drücken als zur Eingabe des Quellen-Codes. Dies sind die Gründe dafür, daß in der FUNKSCHAU manche längeren Programme hexadezimal statt in der leichter verständlichen Assemblerform abgedruckt werden.

Der Abdruck eines Source-Listings allein ist nicht allzu hilfreich, da nicht alle Mikrocomputer-Benutzer über einen Assembler verfügen, jenes Programm also, das die Übersetzung in den Maschinencode besorgt.

```
;.DIESES PROGRAMM
;.DRUCKT VIER 'A'
OUT =$E9BC
MONI =$E182
*=$0200
START LDX #4 ;4MAL
SCHL LDA #'A'
JSR OUT ;AUSGABE
DEX ;NOCHMAL?
BNE SCHL ;JA
JMP MONI ;FERTIG
.END
```

Bild 1. Quellen-Code unseres Beispiel-Programms. Wer keinen Assembler besitzt, müßte diesen Code mühsam von Hand in hexadezimale Operationscodes übersetzen

```
020D          PASS 1
020D          PASS 2
0000
0000          ;DIESES PROGRAMM
0000          ;DRUCKT VIER 'A'.
0000 OUT            =$E9BC
0000 MONI           =$E182
0000                *=$0200
0200 START A204     LDX #4          ;4MAL
0202 SCHL  A941     LDA #'A'
0204       20BCE9   JSR OUT         ;AUSGABE
0207       CA       DEX             ;NOCHMAL?
0208       D0F8     BNE SCHL        ;JA
020A       4C82E1   JMP MONI        ;FERTIG
020D                .END
020D          ERRORS= 0000
```

```
<M>=0200 A2 04 A9 41
< > 0204 20 BC E9 CA
< > 0208 D0 F8 4C 82
< > 020C E1 00 00 00
```

Bild 3. Das ist das eigentliche Maschinenprogramm, wie es der Mikroprozessor versteht – platzsparend, aber unverständlich für die meisten von uns

◀ Bild 2. Assembler-Listing des gleichen Programms, entstanden auf dem Mikrocomputer AIM-65 (CPU = 6502). Dies ist die übersichtlichste und zugleich platzverschwendendste Darstellungsform; von links nach rechts: Adresse, symbolische Adresse, Objektcode, mnemonische Befehle, Kommentare

```
<K>•=0200
/06
 0200 A2 LDX #04
 0202 A9 LDA #41
 0204 20 JSR E9BC
 0207 CA DEX
 0208 D0 BNE 0202
 020A 4C JMP E182
```

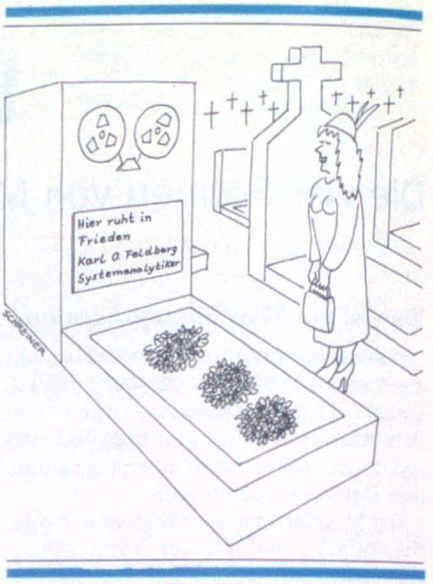
Bild 4. Mit einem Disassembler läßt sich ein headezimaler Speicherinhalt in die leichter verständlichen „Mnemonics" rückübersetzen. Von links nach rechts: Adresse, Operationscode, mnemonischer Befehl, Operand

Programme kapieren – disassemblieren!

Hat man nun aus der FUNKSCHAU einmal ein hexadezimales Programmlisting abgetippt und will wissen, was das Programm eigentlich tut, so kann man sich eines Disassemblers bedienen. Das ist wiederum ein Programm, das genau das Gegenteil eines Assemblers tut: Die im Speicher stehenden Bytes werden in ihre mnemonische Form rückübersetzt und ausgedruckt (*Bild 4*). Der Unterschied zum Assemblerlisting ist allerdings recht deutlich: Natürlich fehlen die sonst recht hilfreichen Kommentarfelder, und es werden keine symbolischen, sondern nur absolute Adressen (d. h. z. B. keine Sprungziel-Namen, sondern die Hex-Adressen) ausgedruckt. Immerhin sieht man mit einem Disassembler aber schon mehr als bei einem reinen Hex-Listing.

Die hier abgedruckten Demonstrationsprogramme entstanden übrigens mit dem Mikrocomputer AIM-65 nebst einem Silent-700-Drucker.

Herwig Feichtinger



4-K-Erweiterung beim KIM-1 im untersten 8-K-Bereich

Beim KIM-1 liegt der vorhandene 1-K-RAM-Bereich auf K0 (0000...3FFF), damit die beiden Pages P0 und P1 benutzt werden können, die beim Mikroprozessor 6502 eine Sonderstellung einnehmen. Der nicht belegte 4-K-Bereich von 0400...13FF kann vom Benutzer mit weiterem Speicher aufgefüllt werden.

Bei den meisten handelsüblichen 4-K-RAM-Platinen muß die Anfangsadresse ein Vielfaches von 4 K sein. Da der freie 4-K-Bereich beim KIM-1 aber auf einer 1-K-Grenze beginnt, ist die Auffüllung dieses Bereiches nicht ohne weiteres möglich. Die Änderung der Adreßdecodierungsschaltung auf der RAM-Platine ist selbst bei genügend Vorkenntnissen noch schwierig.

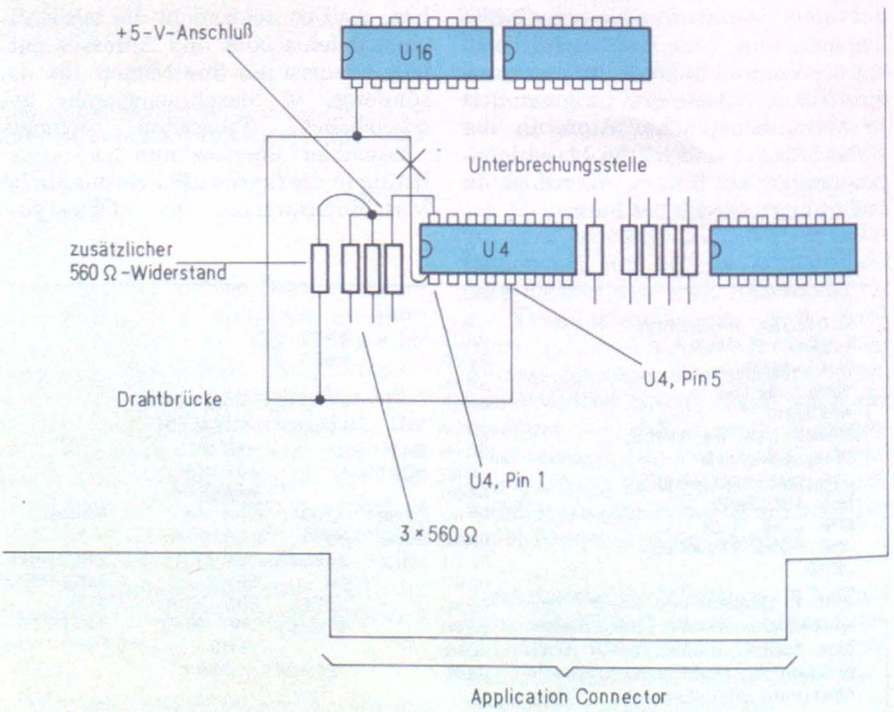
Durch einen sehr einfachen Eingriff auf der KIM-1-Platine kann man dagegen erreichen, daß eine 4-K-Platine beim KIM-1 im unteren 8-K-Bereich (0000...1FFF) auf eine 4-K-Grenze gelegt werden kann. Verlegt man nämlich das bereits vorhandene 1-K-RAM von K0 nach K4 (1000...13FF), so kann man eine 4-K-Platine im Bereich 0000 bis 0FFF anordnen.

Um das vorhandene RAM in den Adreßbereich von K4 zu verlegen, muß die Verbindung von K0 auf U4, Pin 1 (74LS145) zu CS auf Pin 13 der Speicherbausteine U5...U12 bzw. zu Pin 1 des Hex-Inverters U16 (7404) aufgetrennt werden. Statt dessen ist Pin 5 von U4 mit Pin 13 der Speicherbausteine und Pin 1 des Inverters zu verbinden. Da das IC 74LS145 offene Kollektorausgänge besitzt, muß außerdem ein 560-Ω-Widerstand (1/4 W) von U4, Pin 5 zur Versorgungsspannung von + 5 V (z. B. U4, Pin 16) eingelötet werden.

Die Verbindung zwischen U4, Pin 1 und den Anschlüssen Pin 13 auf U5...U12 und Pin 1 auf U16 wird am besten in der Nähe von U4 aufgetrennt. Von U4, Pin 1 führt nämlich direkt eine Leiterbahn zu U16, Pin 1, die einfach durchtrennt werden kann. Dann brauchen nur noch die Drahtverbindung von U4, Pin 5 zu U16, Pin 1 und der 560-Ω-Widerstand eingelötet zu werden. Das *Bild* zeigt die Lage der Unterbrechungsstelle und die Anschlüsse.

Natürlich kann das 1-K-RAM des KIM-1 nur dann verlegt werden, wenn die Pages P0 und P1 wieder von einem anderen RAM-Bereich überdeckt werden.

Dr. Rainer Gerlich



Notwendige Änderungen auf der KIM-1-Platine zur Verschiebung des 1-K-RAM-Bereichs von K0 nach K4

Ein Blick in den Nascom-Monitor

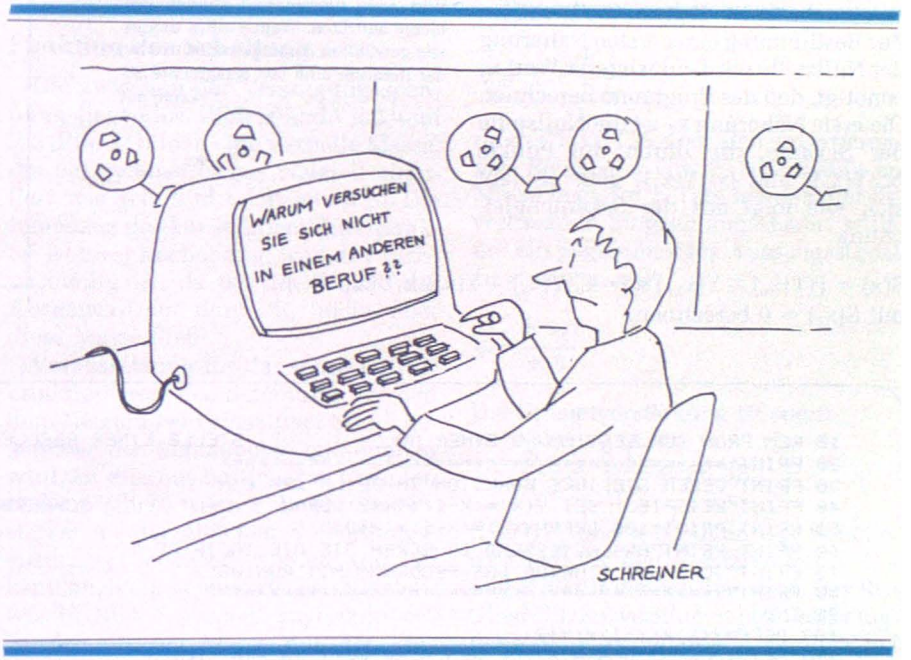
Wer sich entschlossen hat, mit dem Nascom-1 in die Mikroprozessortechnik einzusteigen, dachte dabei sicher an das günstige Preis-Leistungsverhältnis. Für das System kann dies zweifellos gelten, die Softwarebeschreibung reicht dagegen längst nicht an das heran, was bei billigeren Systemen zum Teil geboten wird. Dabei läßt sich aus dem 1-K-Nasbug-Monitorprogramm weit mehr herausholen, als die „Software-Notes" auf den ersten Blick erkennen lassen. Und wer dann noch die kleinen Tücken der verfügbaren Subroutinen kennt, erspart sich eine Reihe Frustrationen und hat bestimmt mehr Spaß an seinen ersten Programmierversuchen.

Das Begleitheftchen macht den Benutzer zunächst einmal auf ein Programm zur Zeitverzögerung aufmerksam. Einmal durchlaufen, soll es den Gang der Dinge um etwa 6 ms verzögern. Wer es bedenkenlos in seine ersten Programme einbaut, wird sich gelegentlich über die Eigenwilligkeit des Computers wundern. Die Ursache ist in dem bewußten Unterprogramm zu suchen. Es löscht nämlich den Akkumulator und verändert heimtückisch die Flagregister. Es sei also empfohlen, vor Aufruf dieser Routine Register A und Flags zu retten.

Dann wird ein Unterprogramm angeboten, das alternierend eine Leuchtdiode ein- und ausschaltet. Dieser Schaltvorgang kann auch zur Steuerung von Kassettenrecordern benutzt werden, wenn man die LED durch ein Relais ersetzt. Bei den Kommandos zur Aufzeichnung oder zum Einlesen von Kassetten wird diese Routine ja bekanntlich benutzt. Auch für sie gilt: Vorher den Akku und das Flagregister retten, da beide verändert werden. In der Beschreibung ist über diese Eigenschaft der beiden Programme nichts zu lesen.

Dann wird die Routine SRLOUT kurz beschrieben. Sie gibt den Akkuinhalt an das UART weiter und tut ein übriges: Das Statusregister des UART wird so lange abgefragt, bis das Datenregister leer ist. Nach der Rückkehr von SRLOUT kann also gleich das nächste Zeichen ausgegeben werden, ohne daß weiterer Programmaufwand nötig ist.

Das Programm KBD fragt das Tastenfeld ab und liefert das Zeichen im Akkumulator an. Wurde eine Taste gedrückt, setzt das Programm das Carryflag, andernfalls ist das Flag rückgesetzt. Auch die anderen Flags können beeinflußt werden, was bei der Anwendung der Subroutine Beachtung finden sollte. Eine weitere Variante stellt CHIN dar: Es fragt sowohl die Tastatur als auch den UART nach Daten ab. Dieses Programm gibt die Kontrolle erst wieder ab, wenn es ein Zeichen gefunden hat. KBD dagegen kehrt nach einmaliger Abfrage der Tastatur wieder zurück. Auch bei CHIN wird das Zeichen im Akku übergeben, die Flags werden ebenfalls beeinflußt.



Mit CRT kann ein im Akku enthaltenes ASCII-Zeichen unter der aktuellen Cursor-Position auf den Bildschirm geschrieben werden. Dieses Programm verändert zwar keines der Datenregister der CPU, die Flags werden dennoch beeinflußt. Will man Einzelzeichen aus einem selbsterstellten Programm oder aus einem Interpreter übergeben, ist diese Routine sehr gut verwertbar. Will man aber einen festen Text ausgeben, geht's viel kürzer: Nach dem RST-5-Befehl (hex EF) kann man den Text im ASCII-Format einfach auflisten und mit hex 00 abschließen. Der Text wird dann von der aktuellen Position des Cursors an ausgegeben, bis ein mit 00 belegter Speicherplatz auftaucht. Dann gibt die mit RST 5 aufgerufene Routine die Kontrolle wieder zurück. Das Monitorprogramm gibt das Zeichen „>" jedesmal auf diese Weise aus. Auch Steuerzeichen können verarbeitet werden, doch ist zu berücksichtigen, daß sie nicht in ASCII, sondern im Nascom-spezifischen Format geschrieben werden müssen ($1D_H$ = Backstep, $1F_H$ = New Line, $1E_H$ = Clear Screen). Akku und Flags fallen dem Programm zum Opfer.

Die Flexibilität des Monitors kommt der Absicht, ein Hexlisting per Drucker zu erstellen, sehr entgegen. Eigentlich ist diese Möglichkeit bereits in dem Kommando „D" für die Ausgabe auf Kassette enthalten. Statt der Kassette einen Drucker anzuschließen, ist leider nicht problemlos möglich, da beim D-Kommande keine Formatsteuerzeichen ausgegeben werden, wie sie für die Druckersteuerung unerläßlich sind. Dennoch läßt sich das Unterprogramm B2HEX verwenden. Dazu ist es nötig, zunächst die RAM-Speicherplätze C4A, C4B und C4C zu ändern. Dort wird bei der Initialisierung vom Monitor ein Sprungbefehl zum Unterprogramm CRT eingeschrieben. Ändert man den Befehl so, daß der Sprung zu SRLOUT erfolgt, bewirkt ein Aufruf der Routine B2HEX (244_H), daß der Inhalt von A in zwei Hexzeichen umgewandelt und seriell ausgegeben wird. Das Listing-Programm, das der Benutzer dann noch zu erstellen hat, kommt dann ganz ohne die Umwandlungs- und Ausgaberoutinen aus und kann sich auf die Formatierung beschränken.

Bernhard Wiedemann

Nullstellen-Errechnung in Basic

Das in *Bild 1* vollständig aufgelistete Basic-Programm errechnet die Nullstellen reeller Funktionen, d.h. eine Zahl x_o mit $y(x_o) = 0$. Dabei müssen folgende Voraussetzungen erfüllt werden:

1. Es ist eine Nullstelle vorhanden.

2. Die Funktion ist in einem genügend großen Intervall, welches die Nullstelle enthält, stetig und definiert.

3. Der einzugebende Startwert liegt genügend nahe bei der Nullstelle.

Zunächst ist die Funktion, deren Nullstelle bestimmt werden soll, nach Programmanweisung zu definieren. Nachdem ein Startwert x_o, der möglichst nahe einer Nullstelle liegen sollte, eingelesen ist, beginnt die Iteration: Zur Bestimmung einer ersten Näherung der Nullstelle wird ein weiterer Wert x_1 benötigt, den das Programm berechnet. Die erste Näherung x_2 ist die Nullstelle der Sekante, die durch die Punkte $(x_o;Y(x_o))$ und $(x_1;Y(x_1))$ geht. x_2 läßt sich wie folgt aus der Sekantengleichung

$S(x) = ((Y(x_o) - Y(x_1))\,x + x_o\,Y(x_1) - Y(x_o)\,x_1)/(x_o - x_1)$

mit $S(x_2) = 0$ berechnen:

$x_2 = (x_1\,Y(x_o) - x_o\,Y(x_1))/(Y(x_o) - Y(x_1))$

Zur Berechnung einer weiteren Näherung x_3 ersetzt man x_o durch x_1 und x_1 durch x_2. x_3 ist die Nullstelle der auf x_1 und x_2 beruhenden Sekante. Man erhält auf diese Weise sukzessive bessere Näherungen für die Nullstelle der Funktion Y(x). Die Iteration wird abgebrochen, sobald der Betrag der Differenz zweier aufeinanderfolgender Näherungen kleiner als 10^{-6} ist.

Falls mit diesem Verfahren keine Nullstelle gefunden wird, so können verschiedene Fälle vorliegen, z.B. die Funktion besitzt keine Nullstelle. Oder der Startwert ist ungünstig gewählt, so daß bei der Iteration eine „fast" waagerechte Sekante auftritt (*Bild 2*). Dann befindet man sich schnell außerhalb des Rechenbereichs (overflow). Möglich ist es auch, daß die Iterationsfolge sich bei einem lokalen Extremum der Funktion festläuft. In diesem Fall wird die weitere Berechnung abgebrochen.

Bernd Schonenberg

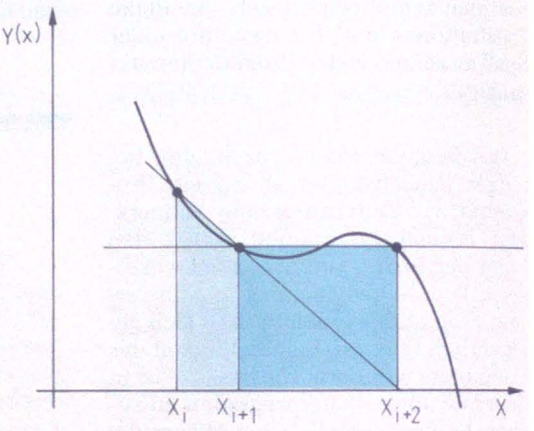


Bild 2. In diesem Fall können Probleme auftreten: Wegen eines ungünstig gewählten Anfangswertes tritt bei der Iteration eine fast waagrechte Sekante auf

```
10 REM PRGM ZUR BESTIMMUNG EINER NULL-          STELLE EINER REELLEN FUNTION
20 PRINT"========================================="
30 PRINT"GEBEN SIE IHRE FUNKTION IN ZEILE 100 EIN"
40 PRINT"BEISPIEL: SEI Y(X)=(X-1)*(X+2),DANN                SCHREIBEN SIE:"
50 PRINT:PRINT"100 DEFFNY(X)=(X-1)*(X+2)"
60 PRINT:PRINT"ANSCHLIESSEND DRUECKEN SIE DIE TASTE RE-"
70 PRINT"TURN UND STARTEN DAS PROGRAMM MIT RUN100"
80 PRINT"========================================="
90 END
100 DEFFNY(X)=(X-1)*(X+2)
110 PRINT:PRINT"GEBEN SIE EINEN STARTWERT X0 EIN, MOEG-"
120 PRINT"LICHST NAHE EINER VERMUTETEN NULLSTELLE"
130 PRINT:INPUT"X0=";X0
140 T=TI
150 N=0: I=6
160 I=I-1: IF ABS(FNY(X0)-FNY(X0+10↑-I))>1E-5 THEN X1=X0+10↑-I:GOTO185
170 IF I>1 THEN 160
180 X1=X0+.1
185 B=FNY(X0)
190 A=B:B=FNY(X1)
200 X2=(X1*A-X0*B)/(A-B):N=N+1
205 IF N>10 THEN IF ABS(X1-X0)<ABS(X2-X1)THEN400
210 IF ABS(X2-X1)<1E-7 THEN XNULL=X2:GOTO300
220 X0=X1:X1=X2:GOTO190
300 PRINT:PRINT:PRINT:PRINT"ANZAHL DER ITERATIONEN";N
305 PRINT:PRINT"SEC:";(TI-T)/60
310 PRINT:PRINT"DIE FUNKTION BESITZT EINE NULLSTELLE BEI"
320 PRINT"XNULL=";X2
330 PRINT:PRINT"FUER WEITERE NULLSTELLEN STARTEN SIE MIT"
340 PRINT"EINEM ANDEREN ANFANGSWERT X0 IN ZEILE100":END
400 PRINT:PRINT:PRINT"MIT DIESEM STARTWERT KEINE NULLSTELLE"
410 PRINT"GEFUNDEN: VERSUCHEN SIE EINEN ANDEREN"
420 PRINT"STARTWERT: PROGRAMMBEGINN: ZEILE 100"
430 PRINT"WAHRSCHEINLICH IST JEDOCH KEINE NULL-"
440 PRINT"STELLE VORHANDEN."
450 PRINT:PRINT"SEC:";(TI-T)/60:END
READY.
```

Bild 1. Vollständiges Programmlisting für die Nullstellenberechnung reeller Funktionen

Ursula und Lutz May

16-KByte-Speicherkarte

2. Teil

Nach der Beschreibung der Schaltung und der Adressenauswahl der statistischen 16-KByte-Speicherkarte wenden wir uns nun dem Zusammenbau zu. Die erforderliche doppelseitige, durchkontaktierte Platine ist von den Verfassern zu haben.

Zusammenbau

Die hier beschriebene Speicherkarte (*Bild 6, Bild 7*) ist beidseitig beschichtet und hat eine hohe Bauteiledichte. Wie bei allen Mikrocomputereinheiten muß auch hier die Karte vor dem Zusammenbau optisch auf eventuelle Produktionsfehler hin überprüft werden. Soll die Speicherkarte nur teilbestückt werden (*Bild 8*), so ist die Verwendung von IC-Sockeln sinnvoll, da sich eine spätere Erweiterung auf diese Weise leichter durchführen läßt. Sollte es sich bei diesem Zusammenbau um eine der ersten Lötarbeiten handeln, so ist eine komplette IC-Sockel-Bestückung unbedingt zu empfehlen, da sich Montage- und Bauteilemängel leichter beheben lassen. Die flachen IC-Sockel von Texas Instruments haben sich gut bewährt. Die Verwendung einer Lötzinn-Saugpumpe ist zu unterlassen, da sie durch ihre ruckende Bewegung die Leiterbahnen zerreißt. Entlötlitze bietet dagegen nicht nur ein mechanisches unproblematisches Arbeiten, sondern auch eine viel größere Saugwirkung bei niedrigeren Löttemperaturen.

Nach dem Einsetzen der IC-Fassungen werden die drei 10-kΩ-Widerstände, die 64polige Steckerleiste und zuletzt die 4 Tantal-Kondensatoren eingelötet. Nun können die einzelnen TTL-

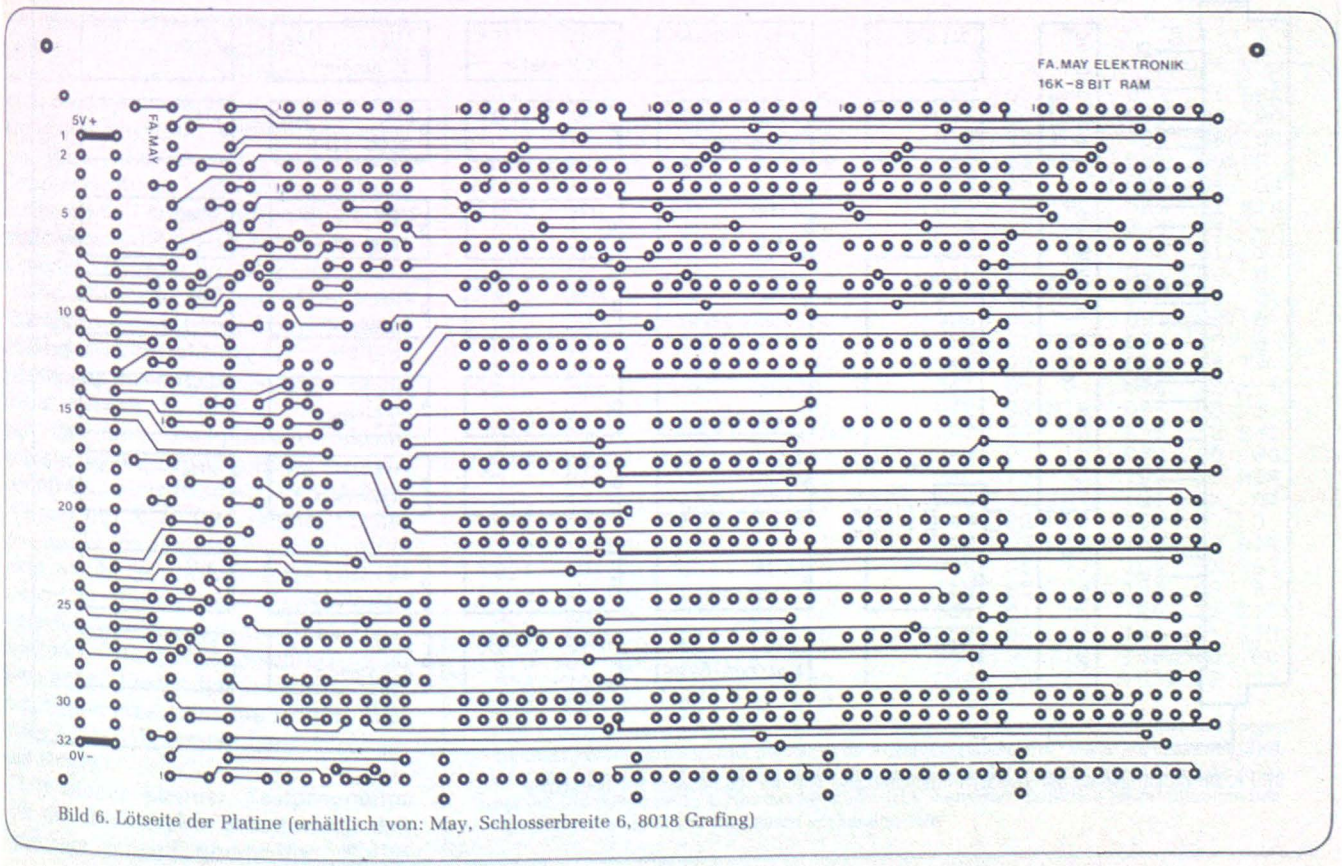


Bild 6. Lötseite der Platine (erhältlich von: May, Schlosserbreite 6, 8018 Grafing)

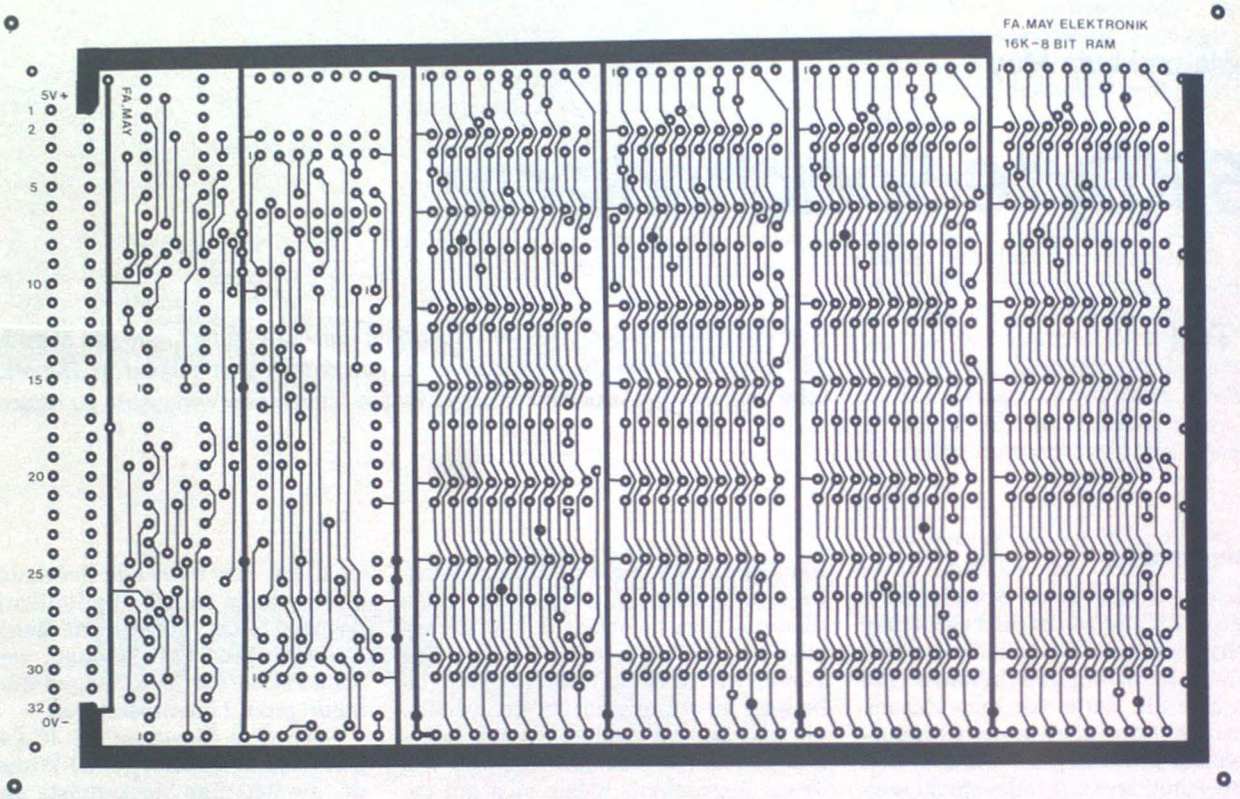


Bild 7. Bestückungsseitige Leiterbahnen der Platine

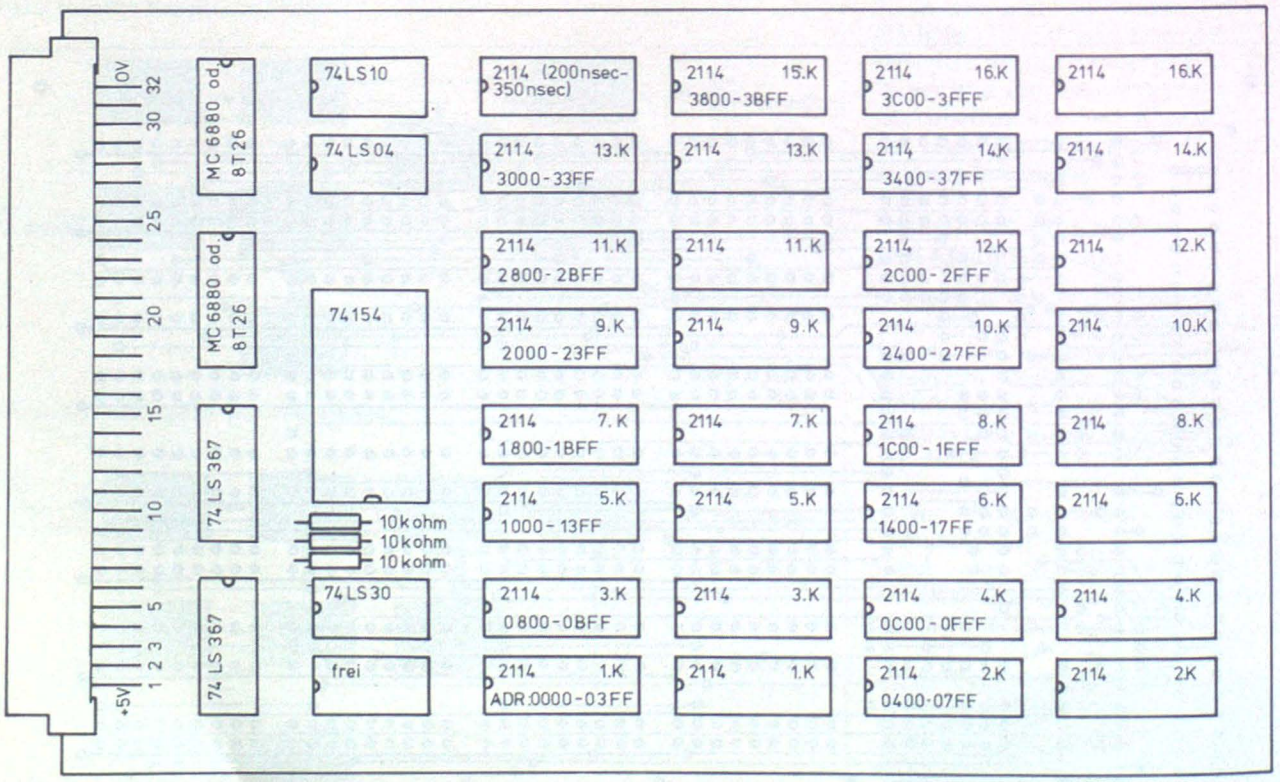


Bild 8. Bestückungsplan der 16-KByte-Speicherkarte von der Bauteilseite her betrachtet

und Speicherelemente in die Fassungen gesteckt werden. Mit einer visuellen Kontrolle können fehlerhafte Lötstellen, z. B. ungewollte Kurzschlüsse oder kalte Lötstellen entdeckt werden.

Bevor die Kartei in Betrieb genommen wird, müssen nun vom Anwender der Adressenbereich nach Tabelle 5 bestimmt und die entsprechenden Lötaugenpaare mit kleinen Drahtbügeln miteinander verbunden werden.

Inbetriebnahme

Mit einem Ohmmeter wird an der bestückten Karte der Widerstand zwischen den beiden Versorgungsanschlüssen (1a und 32a) gemessen. In Abhängigkeit der verwendeten Speichertypen darf hierbei der Wert nicht kleiner als 10 Ω sein. Mit dieser Messung läßt sich ein Kurzschluß innerhalb des Versorgungsnetzes auf der Karte sofort erkennen. Als mögliche Fehlerquellen kommen in Frage:

- verkehrt eingesetzte oder defekte Tantalkondensatoren;
- Kurzschluß durch Lötzinnspritzer oder Lötfehler;
- verkehrt herum eingesetzte TTL- bzw. Speicherbausteine;
- defekter Halbleiterbaustein.

Nach dem Anlegen der Versorgungsspannung (+5 V) wird der Stromverbrauch nach Tabelle 2 gemesen, wobei ein eventuell zu hoher Wert ebenfalls die gleichen Fehlerursachen haben kann wie bei der vorhergenannten Messung. Nach Abschluß dieser Prüfung kann die Speicherkarte mit einem Mikroprozessorsystem verbunden werden. Wird dabei die unter [1] genannte Computerkarte in Verbindung mit der für diesen Fall konstruierten Busleiterplatte verwendet, dann entstehen hier keinerlei Verdrahtungsarbeiten.

Wird die Karte mit einem anderen Mikroprozessorsystem gekoppelt, dann muß bei der Verdrahtung der drei Steuerleitungen besondere Sorgfalt angewandt werden. Es muß sichergestellt sein, daß, bevor der positive Taktimpuls auf der Φ2-Leitung, Bussteckeranschluß 23a, eintrifft, die Daten, Adressen und das R/W-Steuersignal vorliegen; und bevor der positive Taktimpuls nicht wieder in die Low-Lage zurückgekehrt ist, dürfen diese Signale nicht verändert werden. Auf die VMA-Steuerleitung, Bussteckeranschluß 22a, kann unter Umständen verzichtet werden. Dieser Anschluß muß dann über einen 3,3-kΩ-Widerstand auf + 5 V gelegt werden.

Mit einem kleinen Testprogramm läßt sich die korrekte Arbeitsweise des Speichers schnell überprüfen. Sollte jedoch ein Fehler festgestellt werden, dann kann er mit einem Oszilloskop oder einem einfachen Logikanalyser festgestellt werden. Dabei ist folgendes zu bemerken: Die Datenbus-Pufferbausteine IC 37 und 38 invertieren die durchlaufenden Daten so, daß z. B. der hexadezimale Wert 55 als AA im Speicher abgelegt wird. Beim Auslesen bringen die Puffer die Information wieder in die richtige Lage.

Literatur

[1] *May, U.; May, L:* 6802-Computer – selbstgebaut. FUNKSCHAU 1979, Heft 21, S. 1250...1254; Heft 22, S. 1314...1318.

Stichworte zum Inhalt

2114, statisches RAM, Adressendecoder, VMA, Puffer.

Numerisches Drucken mit TI 58/59 und PC 100

Mit den Befehlen PRT und OP 06 lassen sich die 20 Zeichen pro Zeile des Druckers PC 100 nur unvollständig ausnutzen. Dagegen sind mit den Befehlen OP 01 bis OP 05 alle 20 Zeichen jeder Druckzeile zugänglich. Will man auf diese Weise eine Zahl in das Druckregister einbringen, muß sie vorher gemäß der Codetabelle von TI umgewandelt werden. Das Programm in *Bild 1* erledigt diese routinemäßige Umwandlung (Eingabe und Wirkung der HIR-Codes s.u.). Falls erwünscht, erfolgt anschließend auch die Eingabe der umgewandelten Zahl in eines der vier Druckregister.

Beispiel: Die Tastenfolge 876 SBR 065 54321 SBR 071 29 SBR 077 40 +/– SBR 083 OP 05 bewirkt den Ausdruck der Zeile: „ 87654321 (3x) 29 (2x) –40".

Weil dieses Programm außer den Hilfsregistern 1 bis 8 (s. u.) und dem t-Register keine Speicher benutzt, kann man es problemlos als Unterprogramm in eigenen Programmen verwenden. Besonders nützlich ist das Umwandlungsprogramm bei der Ausgabe von Tabellen. Beispiel: Das Programm in *Bild 2* (Start durch Drücken der Taste A) bewirkt den Ausdruck der Tabelle in *Bild 3* für x (im Gradmaß) und $10^{13} \cdot \sin(x)$, aus der alle 13 intern berechneten Stellen der Sinusfunktion abzulesen sind.

Die Eingabe eines beliebigen Codes wie z.B. 82 für den HIR-Befehl im Um-

000	29	CP	055	00	00	028	69	DP	083	71	SBR
001	22	INV	056	32	32	029	20	20	084	00	00
002	67	EQ	057	25	CLR	030	22	INV	085	00	00
003	00	00	058	82	HIR	031	28	LOG	086	69	DP
004	07	07	059	14	14	032	85	+	087	04	04
005	01	1	060	42	STO	033	22	INV	088	92	RTN
006	92	RTN	061	00	00	034	77	GE	089	55	÷
007	32	X:T	062	82	HIR	035	00	00	090	01	1
008	22	INV	063	13	13	036	39	39	091	00	0
009	77	GE	064	92	RTN	037	02	2	092	00	0
010	00	00	065	71	SBR	038	85	+	093	00	0
011	14	14	066	00	00	039	01	1	094	00	0
012	02	2	067	00	00	040	00	0	095	95	=
013	00	0	068	69	DP	041	00	0	096	32	X:T
014	82	HIR	069	01	01	042	82	HIR	097	05	5
015	03	03	070	92	RTN	043	43	43	098	48	EXC
016	09	9	071	71	SBR	044	01	1	099	00	00
017	32	X:T	072	00	00	045	75	-	100	82	HIR
018	50	I×I	073	00	00	046	59	INT	101	04	04
019	55	÷	074	69	DP	047	82	HIR	102	00	0
020	28	LOG	075	02	02	048	33	33	103	82	HIR
021	59	INT	076	92	RTN	049	95	=	104	03	03
022	48	EXC	077	71	SBR	050	65	×	105	09	9
023	00	00	078	00	00	051	01	1	106	32	X:T
024	82	HIR	079	00	00	052	00	0	107	61	GTO
025	04	04	080	69	DP	053	97	DSZ	108	00	00
026	43	RCL	081	03	03	054	00	00	109	32	32
027	00	00	082	92	RTN						

Bild 1. Umwandlungsprogramm. Die Umwandlung selbst kann einzeln mit SBR 000 vorgenommen werden und wirkt auf die Zahl a im Anzeigeregister für $10^5 > a > -10^4$. SBR 065, SBR 071, SBR 077 und SBR 083 bewirken die Umwandlung und die anschließende Eingabe in eines der Druckregister 1...4. Nur das t-Register wird verändert. Es dürfen weder unvollendete Operationen noch offene Klammern vorhanden sein

```
110  76 LBL   151  00  0      131  00  0      172  00  0
111  11  A    152  95  =      132  00  0      173  95  =
112  60 DEG   153  42 STO      133  95  =      174  71 SBR
113  09  9    154  03  03     134  42 STO      175  00  00
114  42 STO   155  59 INT      135  03  03     176  89  89
115  01  01   156  22 INV      136  59 INT      177  69 OP
116  01  1    157  44 SUM      137  22 INV      178  04  04
117  00  0    158  03  03     138  44 SUM      179  69 OP
118  42 STO   159  71 SBR      139  03  03     180  05  05
119  02  02   160  00  00     140  71 SBR      181  01  1
120  43 RCL   161  89  89     141  00  00     182  00  0
121  02  02   162  69 OP       142  71  71     183  44 SUM
122  71 SBR   163  03  03     143  43 RCL      184  02  02
123  00  00   164  43 RCL      144  03  03     185  97 DSZ
124  65  65   165  03  03     145  65  ×      186  01  01
125  43 RCL   166  65  ×      146  01  1      187  01  01
126  02  02   167  01  1      147  00  0      188  20  20
127  38 SIN   168  00  0      148  00  0      189  25 CLR
128  65  ×    169  00  0      149  00  0      190  91 R/S
129  01  1    170  00  0      150  00  0
130  00  0    171  00  0
```

Bild 2. Beispielprogramm zum Ausdruck der Tabelle im Bild 3. Start durch Drücken der Taste A

wandlungsprogramm ist möglich durch Eingabe von z.B. STO 82 und anschließendes Löschen des STO durch den DEL-Befehl. Die HIR-Befehle, deren Existenz von der Firma TI in dem Handbuch für die Rechner TI 58/59 leider nicht erwähnt wird, wirken auf die Hilfsregister (auch Hire- oder Hierarchie-Register genannt). Diese Hilfsregister (8 Stück, numeriert von 1 bis 8) werden normalerweise durch unvollständige Operationen belegt. Die letzten 4, also Nr. 5 bis 8, sind die Druckregister 1 bis 4. Der Befehl HIR xy wirkt für y = 1,2...8 auf die Anzeige und das Hilfsregister y, wie die Tabelle in *Bild 4* angibt. Für x = 2 oder y = 0 bzw. y = 9 scheinen die Befehle HIR xy keine Wirkung zu haben, bis auf HIR 20, das bei einigen Statistikfunktionen intern verwendet wird. Die (wirksamen) Befehle HIR xy haben für x ≧ 3 leider eine unangenehme Nebenwirkung: ist die Zahl a im Anzeigeregister nicht im Exponential- oder Eng-Format gegeben, so wird für |a| < 1 der interne Exponent von a positiv. Beispielsweise wird 0.123 umgewandelt in 12.3 und 0.024 in 240. Bei unseren Anwendungen spielt das aber keine Rolle, da nur ganze Zahlen verarbeitet werden.

Mit den obigen beiden Beispielen sind die Möglichkeiten der Eingabe in die Druckregister noch nicht erschöpft; wir können nachträglich noch Zeichen einfügen oder löschen. Als Beispiel nehmen wir an, daß die letzte Druckzeile aus der Tabelle im Bild 3 noch gespeichert ist. Wir wollen nun in das 1. Druckregister an den beiden ersten Stellen noch die Ziffern 35 einfügen. Das Programm in *Bild 5* bewirkt diese Einfügung. Zunächst wird dabei die Zahl 35 mit SBR 000 umgewandelt. Da der Befehl OP 01 eine ganze Zahl abcdefghijk in der Form 0.00abcdefghijk in das Druckregister einträgt, wird die umgewandelte Zahl in den Programmschritten 198 bis 201 mit dem richtigen Exponenten versehen. Die Addition der 1 in den Schritten 202 bis 204 ist erforderlich, damit nach dem Addieren im Druckregister (Schritte 205 und 206) die Ziffern nicht nach links verschoben werden; durch OP 05 werden ja die letzten 10 der 13 vorhandenen Ziffern im Druckregister ausgedruckt.

Dr. Harald Lindner

```
10   1736481776662
20   3420201433248
30   5000000000000
40   6427876096862
50   7660444431187
60   8660254037858
70   9396926207872
80   9848077530126
90  10000000000000
```

Bild 3. Druckbeispiel mit 13stelliger Ausgabe; die letzten Stellen der Sinusfunktion sind natürlich nicht immer ganz richtig

```
x  HIR xy

0  STO
1  RCL
3  SUM
4  PRD
5  INV SUM
6-9 INV PRD
```

Bild 4. Wirkungsweise der HIR-Befehle (y = 1, 2, ...8 ist die Nummer des Hilfsregisters)

```
191  76 LBL   202  85  +
192  12  B    203  01  1
193  03  3    204  95  =
194  05  5    205  82 HIR
195  71 SBR   206  35  35
196  00  00   207  69 OP
197  00  00   208  05  05
198  55  ÷    209  25 CLR
199  01  1    210  22 INV
200  52 EE    211  52 EE
201  06  6    212  91 R/S
```

Bild 5. Programmbeispiel zum Einfügen von Zeichen in ein Druckregister, ohne dieses zu löschen. Start durch Drükken der Taste B

µC-Nachrichten

Der Hofacker-Verlag vertreibt die im April/Mai 1980 erstmals erschienene Zeitschrift **„Single-Board Computer"**, die sich speziell mit Einplatinen-Computern und der Programmierung in den Maschinencodes der CPUs, 6502 und 1802 befaßt. Im Vordergrund stehen Systeme wie KIM, SYM, AIM, Superboard und 1802-Platinen. Leider steht dem US-Preis von 2 Dollar der deutsche Preis von 9.80 DM gegenüber – so schlecht steht der Dollar offenbar gar nicht...

□

Die Holzkirchener Firma Redysoft vertreibt einen **Level-II-Basic-Compiler** für den TRS-80. Die Ausführungszeit vieler Programme verkürzt sich damit um rund eine Zehnerpotenz, insbesondere bei Verwendung von GOTO, GOSUB und Ausdrücke mit Integer-Variablen. Redysoft ist übrigens ein „Ableger" von Ing. W. Hofacker.

□

Rodnay Zaks schrieb ein Buch namens **„6502 games"**, das der Hofakker-Verlag vertreibt und zahlreiche Spielprogramme in 6502-Maschinensprache enthält. Die Adressenbelegungen entsprechen den Mikrocomputern KIM, SYM und AIM. Die Programme sind recht gut beschrieben, so daß sich das Buch auch zur Vertiefung der Programmierkenntnisse eignet.

□

Rockwell brachte für AIM-65-Besitzer eine englischsprachige Zeitschrift namens **„Interactive"** heraus, die von Eric Rehnke gestaltet wird. Die rund 10 000 AIM-Besitzer in Europa erhalten sie durch Einsenden der gelben Erfassungskarte, die jedem AIM-65 beiliegt. Anfragen sind an System Kontakt, Siemensstr. 5, 7107 Bad Friedrichshall, zu richten.

□

In Braunschweig gibt es nun eine **Arbeitsgruppe Mikrocomputer**, die vom VDE und von der TU Braunschweig getragen wird. Ingenieure und Studenten dieser Arbeitsgruppe pflegen den Erfahrungs- und Softwareaustausch. Interessenten können sich an den VDE, Stresemannallee 21, 6000 Frankfurt/M., wenden.

AIM-65 als Duplex-Terminal

Mit dem hier vorgestellten 6502-Maschinenprogramm ist es möglich, die Mikrocomputer AIM-65 (Rockwell) und PC-100 (Siemens) als vollwertige ASCII-Terminals zu verwenden. Da der Empfang mit einer Interrupt-Routine erfolgt, ist gleichzeitiges Senden und Empfangen (Duplexbetrieb) möglich [1].

Als Schnittstelle zur Außenwelt dienen die TTY-Anschlüsse des AIM-65. An sie kann z. B. ein Telefon-Modem [2] angeschlossen werden, wobei lediglich noch eine Pegelanpassung [3] erforderlich ist. Zu Testzwecken kann auch ein normales Terminal dienen. Der AIM-Schalter „TTY" (S 3) bleibt in Stellung KB.

Das Programm im *Bild* besteht aus dem Hauptteil (Startadresse 0200 bzw. Taste F1) und einer Interrupt-Routine, die periodisch angesprungen wird und dazu dient, Zeichen von der TTY-Schnittstelle in einen FIFO-Speicher einzulesen. Der FIFO beginnt bei der Adresse 0170 und umfaßt 100 Zeichen. Er sorgt dafür, daß beim Empfang keine Totzeiten auftreten, wenn z. B. der Mikrocomputer mit der Ausgabe auf den eingebauten Thermodrucker oder mit dem Senden eines Zeichens beschäftigt ist. Da der Empfang vollständig in der Interrupt-Routine erfolgt, kann unabhängig voneinander gesendet und empfangen werden.

Die Bedienung

Beim Start des Programms wird der Benutzer zunächst gefragt, ob „AUTO-LF" gewünscht werde[1]). Darunter versteht man, daß beim Drücken der Return-Taste nicht nur ein Return-, sondern auch ein Line-Feed-Zeichen gesendet wird (LF = CTRL J).

Dann fragt der Mikrocomputer, ob Duplexbetrieb gewünscht wird. Wenn diese Frage mit „N" beantwortet wird, so erscheinen auch die eingetippten und gesendeten Zeichen später im Display; andernfalls sind nur die empfangenen Zeichen zu sehen. Der Duplex-Modus ist immer dann sinnvoll, wenn man mit einem Computer verkehrt, der alle Zeichen „echot".

[1]) Sollte in seltenen Fällen das Display beim Start dunkel bleiben, so ist vorher in die Zelle A417 ein Wert ungleich Null einzuschreiben, da sich sonst die Interrupt-Routine „aufhängt", wenn das Programm bei offenem TTY-Eingang gestartet wird.

Schließlich will der Mikrocomputer noch die gewünschte Baudrate wissen; mit den Tasten A oder B kann man 110 Bd oder 300 Bd wählen. Ein Parity-Bit wird grundsätzlich nicht ausgestrahlt; stattdessen liegt das achte gesendete Bit stets auf log. 0.

Nach Beantwortung dieser drei Fragen ist das Terminal betriebsbereit. Will man während des Betriebes einen der Parameter ändern, so genügt ein Druck auf die Escape-Taste, um alle drei neu eingeben zu können. Ferner ist es wie gewohnt möglich, den AIM-Drucker mit CTRL PRINT ein- und auszuschalten. Besonderer Trick: Während der Parameter-Eingabe werden weiterhin Zeichen empfangen und im FIFO zwischengespeichert, so daß sie nicht verlorengehen.

Korrekturen sind mit CTRL H und DEL möglich; dabei muß man sich natürlich nach den Erfordernissen der Gegenstation (z. B. des angeschlossenen Rechners) richten. Der AIM-65 begrenzt die Zeilenbreite auf 60 Zeichen; man darf also nicht vergessen, rechtzeitig Return zu drücken. Die Erzeugung und Verarbeitung von Kleinbuchstaben ist grundsätzlich nicht möglich. Drückt man eine Taste länger als ca. 0,3 s, wird sie automatisch wiederholt. Das Programm ermöglicht zusammen mit dem erwähnten Modem nicht nur den Verkehr mit Großrechnern über das Telefonnetz, sondern auch ASCII-Funkfernschreiben. Im Prinzip sind auch höhere Geschwindigkeiten als 300 Bd möglich; dazu wären die Daten an den Adressen 0364...0367 entsprechend den Angaben im AIM-Handbuch zu ändern.

Ein paar Worte zum Programm

Das Programm beinhaltet einige erwähnenswerte Besonderheiten. Es läuft normalerweise während des Terminal-Betriebes in der Schleife zwischen 023D und 0296. In diesem Teil erfolgt das Einlesen der Tastatur-Zeichen, ihre Ausgabe auf die serielle TTY-Schnittstelle und die Anzeige von Zeichen aus dem FIFO.

Die Schwierigkeit ist nun, daß die Tastaturabfrage etwas anders als bei den Routinen erfolgen muß, die als Monitor-Unterprogramm zur Verfügung stehen, denn das Programm muß ja auch dann weiterlaufen, wenn gerade keine Taste gedrückt ist. Zwar könnte das mit dem Monitor-Unterprogramm ROONEK abgefragt werden, doch betrachtet dieses Programm auch Shift und CTRL als vollwertige Tasten und würde die Terminal-Schleife anhalten, wenn z. B. nur Shift gedrückt wurde.

Dies läßt sich sehr leicht am Basic-Interpreter des AIM-65 zeigen: Enthält das Basic-Programm einen GET-Befehl, so bleibt es völlig stehen, wenn kurz Shift oder CTRL gedrückt wird – eine Eigenschaft, die kaum bekannt ist und darauf zurückzuführen ist, daß sich die Firma Microsoft die Basic-Entwicklung des AIM wohl etwas zu einfach gemacht hat.

Unser Terminal-Programm enthält daher eine eigene Routine, die feststellt, ob eine Taste gedrückt ist und dabei Shift und CTRL einfach ignoriert. Ist keine Taste gedrückt, so setzt diese Routine (CKB) das Flag FLG auf Null. Dies wird vom Hauptprogramm als Erlaubnis gewertet, den nachfolgenden Tastendruck auszuwerten und FLG zu erhöhen. Bleibt die Taste gedrückt, so wird sie – da FLG nun ungleich Null ist – nicht mehr ausgewertet, weil ein Tastendruck ja nur ein Zeichen erzeugen darf. Da FLG aber erst nach einer Maskierung mit hex 1F auf Null geprüft wird, erkennt es das Hauptprogramm auch als Null, wenn es 32mal inkrementiert wurde. Daraus ergibt sich eine automatische Wiederholfunktion, wenn man eine Taste länger als etwa 0,3 s drückt.

Bei Duplexbetrieb werden die eingetippten Zeichen zwar zunächst durchaus auf das Display ausgegeben, aber mit dem BS-Unterprogramm sofort wieder gelöscht. Bei Halbduplex-Be-

```
010F          PASS 1
010F          PASS 2
0000
0000          ;AIM AS DUPLEX TERMINAL
0000 IER              =$A00E        ;6522
0000 ACR              =$A00B
0000 T1L              =$A004
0000 T1H              =$A005
0000 INP              =$A800
0000 KBA              =$A480        ;6532
0000 KBB              =$A482
0000 IRQV             =$A404        ;MONRAM
0000 CURP             =$A415
0000 BDH              =$A417
0000 DEB              =$ED2A        ;MONROM
0000 OUT              =$EEFC
0000 CRLF             =$E9F0
0000 PRT              =$F000
0000 PSLS             =$E7DC
0000 TTY              =$EEA8
0000 KEY              =$E93C
0000 GET              =$EC46
0000 BUF              =$170         ;FIFO
0000 CHR              =$FF          ;ZPAGE
0000 BITS             =$FE
0000 PNT1             =$FD
0000 PNT2             =$FC
0000 DUFL             =$FB
0000 LFFL             =$FA
0000 FLG              =$F9
0000                  *=$200
0200 START  A9C0      LDA #$C0      ;INIT
0202        8D0EA0    STA IER       ;VIA
0205        A200      LDX #0
0207        8E0BA0    STX ACR
020A        86FD      STX PNT1      ;AND
020C        86FC      STX PNT2      ;FIFO
020E        A9DC      LDA #<IRQ
0210        8D04A4    STA IRQV
0213        A902      LDA #>IRQ
0215        8D05A4    STA IRQV+1
0218        8D05A0    STA T1H
021B        58        CLI
021C WARM   20AC02    JSR PRI       ;ASK
021F        85FA      STA LFFL      ;USER
0221        20AC02    JSR PRI       ;FOR
0224        85FB      STA DUFL      ;AUTO LF,
0226        20AC02    JSR PRI       ;DUPLEX,
0229        2902      AND #2        ;BAUD
022B        AA        TAX           ;RATE
022C        BD6403    LDA BDR,X
022F        85F9      STA FLG       ;INIT
0231        8D17A4    STA BDH       ;FLG>0
0234        BD6503    LDA BDR+1,X
0237        8D18A4    STA BDH+1
023A        20F0E9    JSR CRLF
023D LOOP   20BC02    JSR CKB
0240        9039      BCC NOKY
0242        20BC02    JSR CKB
0245        9034      BCC NOKY
0247        A5F9      LDA FLG
0249        291F      AND #$1F      ;REPEAT
024B        D02C      BNE RPT       ;KEY
024D        2046EC    JSR GET
0250        C91B      CMP #$1B      ;ESC.?
0252        F0C8      BEQ WARM
0254 OK     48        PHA           ;TO
0255        2000F0    JSR PRT       ;PRTR
0258        20A8EE    JSR TTY       ;+TTY
025B        68        PLA
025C        C90D      CMP #$D
025E        D00D      BNE M1
0260        A5FA      LDA LFFL
0262        C94E      CMP #'N
0264        F015      BEQ NOKY
0266        A90A      LDA #$A       ;ADD
0268        20A8EE    JSR TTY       ;L.F.
026B        D00E      BNE NOKY
026D M1     A4FB      LDY DUFL
026F        C04E      CPY #'N ;IF
0271        F003      BEQ *+5       ;DUPLEX,
0273        20A402    JSR BS        ;CURSOR
0276        209902    JSR RUB       ;BACK
0279 RPT    E6F9      INC FLG
027B NOKY   A5FD      LDA PNT1      ;TEST
027D        C5FC      CMP PNT2      ;FIFO
027F        F0BC      BEQ LOOP
0281        E6FC      INC PNT2
0283        A4FC      LDY PNT2
0285        C064      CPY #100
0287        D004      BNE *+6
0289        A000      LDY #0
028B        84FC      STY PNT2
028D        B97001    LDA BUF,Y     ;PRINT
0290        20FCEE    JSR OUT       ;CHAR.
0293        209902    JSR RUB
0296        4C3D02    JMP LOOP
0299 RUB    C97F      CMP #$7F      ;TEST
029B        F004      BEQ *+6       ;IF BS/
029D        C908      CMP #8        ;RUBOUT
029F        D01A      BNE M2
02A1        20A402    JSR BS        ;YES
02A4 BS     AC15A4    LDY CURP
02A7        F012      BEQ M2
02A9        4CDCE7    JMP PSLS
02AC PRI    BD3603    LDA TXT,X     ;PRINT
02AF        F006      BEQ PRI1      ;TEXT
02B1        20FCEE    JSR OUT
02B4        E8        INX
02B5        D0F5      BNE PRI
02B7 PRI1   E8        INX
02B8        203CE9    JSR KEY
02BB M2     60        RTS
02BC CKB    202AED    JSR DEB       ;TEST
02BF        A9F6      LDA #$F6      ;IF KEY
02C1        8D80A4    STA KBA       ;PRESSED
02C4        AC82A4    LDY KBB
02C7        C8        INY           ;IGNORE
02C8        D00C      BNE KY        ;SHIFT/
02CA        8C80A4    STY KBA       ;CTRL
02CD        AD82A4    LDA KBB
02D0        29FE      AND #$FE
02D2        C9FE      CMP #$FE
02D4        F002      BEQ NK
02D6 KY     38        SEC
02D7        60        RTS
02D8 NK     18        CLC
02D9        84F9      STY FLG       ;FLG=0
02DB        60        RTS
02DC IRQ    48        PHA           ;SAVE
02DD        98        TYA           ;Y & A
02DE        48        PHA
02DF        A5FE      LDA BITS
02E1        D016      BNE NXT
02E3        2C00A8    BIT INP       ;START
02E6        5004      BVC STRT      ;BIT?
02E8        A901      LDA #1
02EA        D03D      BNE RET+3
02EC STRT   A909      LDA #9        ;YES
02EE        85FE      STA BITS
02F0        AD17A4    LDA BDH       ;DELAY
02F3        4A        LSR A         ;1.5
02F4        6D17A4    ADC BDH       ;BITS
02F7        D030      BNE RET+3
02F9 NXT    C6FE      DEC BITS
02FB        F00A      BEQ RDY
02FD        AD00A8    LDA INP       ;READ
0300        0A        ASL A         ;BIT
0301        0A        ASL A
0302        66FF      ROR CHR
0304        4C2603    JMP RET
0307 RDY    A5FF      LDA CHR       ;IGNORE
0309        297F      AND #$7F      ;PARITY
030B        C908      CMP #8        ;AND
030D        F008      BEQ LEG       ;CTRL
030F        C90D      CMP #$D       ;CHR'S
0311        F004      BEQ LEG
0313        C920      CMP #$20
0315        300F      BMI RET
0317 LEG    E6FD      INC PNT1      ;CHAR
0319        A4FD      LDY PNT1      ;INTO
031B        C064      CPY #100      ;FIFO
031D        D004      BNE *+6
031F        A000      LDY #0
0321        84FD      STY PNT1
0323        997001    STA BUF,Y
0326 RET    AD17A4    LDA BDH       ;START
0329        AC18A4    LDY BDH+1     ;TIMER
032C        8C04A0    STY T1L
032F        8D05A0    STA T1H
0332        68        PLA           ;RESTORE
0333        A8        TAY           ;Y & A
0334        68        PLA
0335        40        RTI
0336 TXT    0D        .BYT $D,'AUTO'
0337        4155
033B        2D4C      .BYT '-LF:Y/N?',0
0343        00
0344        0D        .BYT $D,'DUPLEX:'
0345        4455
034C        592F      .BYT 'Y/N?',0
0350        00
0351        0D        .BYT $D,'BAUD: A='
0352        4241
035A        3131      .BYT '110,B=300',0
0363        00
0364 BDR    23        .BYT $23,$3F
0365        3F
0366        0C        .BYT $C,$C2
0367        C2
0368                  *=$10C
010C        4C0002    JMP START     ;<F1>
010F                  .END
010F           ERRORS= 0000

  ROCKWELL AIM 65

<M>=0336 0D 41 55 54
< > 033A 4F 2D 4C 46
< > 033E 3A 59 2F 4E
< > 0342 3F 00 0D 44
< > 0346 55 50 4C 45
< > 034A 58 3A 59 2F
< > 034E 4E 3F 00 0D
< > 0352 42 41 55 44
< > 0356 3A 20 41 3D
< > 035A 31 31 30 2C
< > 035E 42 3D 33 30
< > 0362 30 00 23 3F
<
```

AIM-65-Assemblerlisting für das Terminal-Programm. Der Speicherinhalt für die Texte ist getrennt aufgeführt, da der Assembler nur jeweils die ersten zwei Buchstaben als Hex-Code ausdruckt

trieb werden die Zeichen normal auf das Display geschrieben; eine Besonderheit betrifft hier die Backspace-Funktion (CTRL H bzw. DEL): Diese Steuerzeichen rücken den (nicht sichtbaren) Cursor um eine Position nach rechts. Infolgedessen muß er mit dem Unterprogramm PSLS zweimal nach links verschoben werden, um das zuletzt eingegebene Zeichen zu löschen.

Wie man erkennt, war zum Teil eine etwas trickreiche Programmierung notwendig, um die eine oder andere Schwäche des AIM-Monitorprogramms auszugleichen. Trotzdem zeigt das Programm, wie komfortabel die Möglichkeiten sind, die die AIM-65-Hardware speziell mit den Interrupt-Timern bietet.

Herwig Feichtinger

Literatur

[1] Anwendungsbeispiele für den Mikroprozessor 6502, Seite 43. RPB 173, Franzis-Verlag, München.

[2] Telefon-Modem. FUNKSCHAU 1980, Heft 10, Seite 105.

[3] Noch einmal: Telefon-Modem. FUNKSCHAU 1980, Heft 11, Seite 108.

Pulsmessung per Computer

Computer können durchaus zur Gesundheit – zumindest zu deren Überprüfung – beitragen. Für die Messung der menschlichen Pulsfrequenz genügt ein kleiner Hardware-Zusatz, der zur Umsetzung des Pulses in Digitalimpulse dient.

Zur Messung der Pulsfrequenz dient hier eine Lichtschranke, bestehend aus einem Glühbirnchen und einem Fotowiderstand. Bringt man eine Fingerkuppe in den Strahlengang, so ändert sich die Lichtdurchlässigkeit in Abhängigkeit vom Blutdurchfluß, so daß sich der Puls elektrisch messen läßt.

Diesem Zweck dient die in *Bild 1* wiedergegebene Schaltung. Sie benutzt einen Vierfach-Operationsverstärker LM 324, um die schwachen Änderungen am Fotowiderstand in ein vom Computer verwertbares Digitalsignal umzuformen. *Bild 2* zeigt ein geeigne-

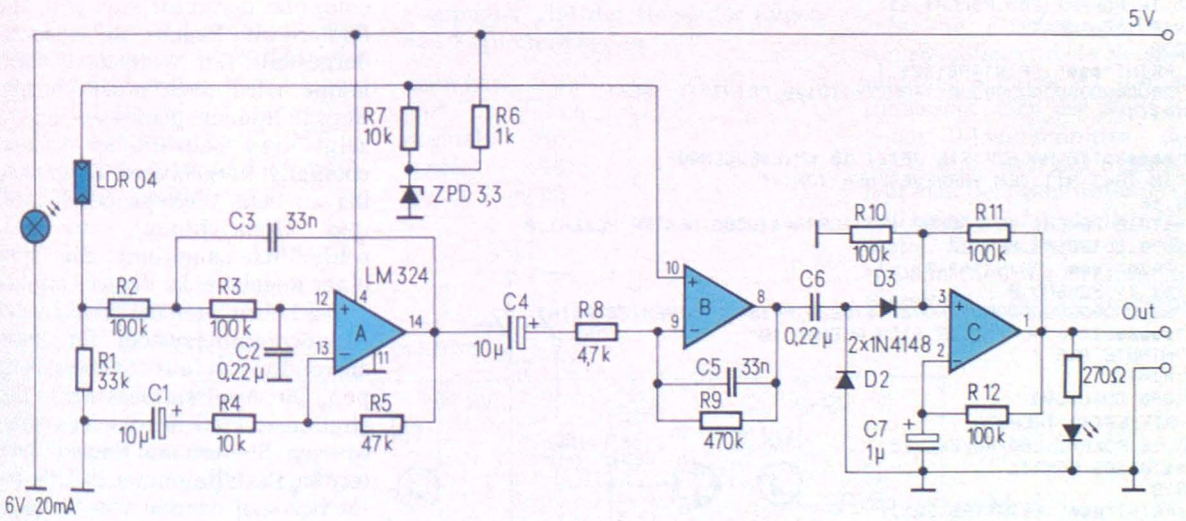
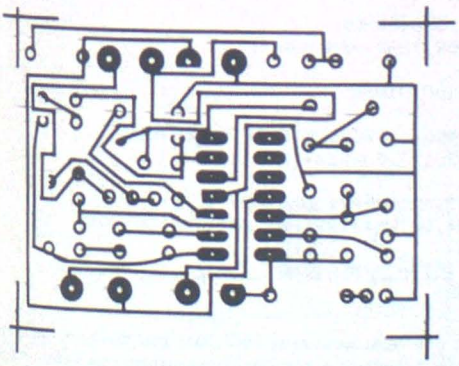


Bild 1. Schaltung zur Umwandlung von Lichtschwankungen in Digitalimpulse. Zwischen Glühbirnchen und LDR ist für die Pulsmessung z. B. der Zeigefinger zu halten, der in Abhängigkeit vom Blutfluß seine Lichtdurchlässigkeit ändert



◄ Bild 2. Passendes Platinen-Layout für die Schaltung in Bild 1

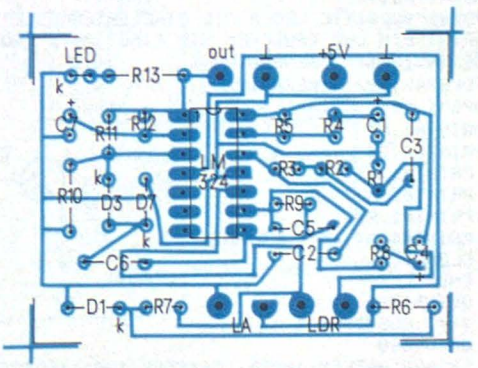


Bild 3. ► Bestückungsplan. Der LM 324 ist ein Vierfach-Operationsverstärker

```
280 PRINT"♥QQMOECHTEN SIE DEN PULS MESSEN (=1)Q"
290 PRINT"ODER VON BAND EINLESEN (=2) ? "
300 GETT$:IFT$<>"1"ANDT$<>"2"THEN300
310 IFT$="2"THENRUN1130
320 INPUT"♥QQQWIE HEISSEN SIE  ";N$
330 DATA169,0,141,232,3,141,233,3,141,234,3,169,127,141,67,232,173,79
340 DATA232,240,251,173,79,232,208,251,173,79,232,208,16,238,232,3,208
350 DATA246,238,233,3,208,241,238,234,3,76,84,3,173,79,232,240,16,238
355 DATA  232 ,3,208,246,238,233,3,208,241,238,234,3,76,97,3,96
360 FORI=826TO894:READA:POKEI,A:NEXTI
370 DATA165,180,181,161,182,170,167
380 DATA59466,59464,59467
390 FORI=1TO7:READL(I):NEXTI
400 READH1:READH2:READH3
410 INPUT"QQWELCHES DATUM HABEN WIR HEUTE";D$
420 A1$="  ____________________"
430 A2$="  |                   | "
440 A3$="  ____________________"
450 PRINT"♥     AKTUELLER PULSQ"
460 PRINTA1$:PRINTA2$:PRINTA3$:PRINT"QQ"
470 PRINTA1$:PRINTA2$:PRINTA3$:PRINT"QQ"
480 PRINTA1$:PRINTA2$:PRINTA3$
490 GOTO610
500 PRINT"S    S";I;"S       ";
510
520 FORY=1TOYY:PRINT"Q";:NEXT
530 H=I-40
540 IFH<8THEN560
550 PRINT"R _";:H=H-8:GOTO540
560 IFH>4THENPRINTCHR$(18);
570 PRINTCHR$(L(H));"_";
580 FORK=1TO24.9-I/8:PRINT" ";:NEXTK
590 PRINT:PRINT"●●":PRINTTAB(27);"| "
600 RETURN
610 PRINT"QQBITTE RUHEN SIE SICH ZUNAECHSTQ"
620 PRINT"1 MINUTE AUS ! ":A1=TI
630 YY=3:GOSUB1090:GOSUB500
640 IF(TI-A1)/60<60 THEN630
650 POKEH3,16:POKEH2,200:POKEH1,15
660 FORI1=1TO1500:NEXTI1
670 POKEH3,0
680 PRINT:PRINT"●●●":PRINTTAB(32);I
690 PRINT"SQQQQQQQQQQQQQQQQQQ":FORZZ=1TO120:PRINT" ";:NEXT
700 W1$=STR$(I)
710 YY=9
720 PRINT"●●●●BITTE MACHEN SIE JETZT 10 KNIEBEUGENQQ"
730 PRINT"IM TAKT MIT DEM VORGEGEBENEN TON !"
740 POKEH3,16
750 FORZZ=1TO10:POKEH1,15:POKEH2,100:FORRR=1TO500:NEXTRR:POKEH1,0
760 GOSUB1090:GOSUB500:NEXTZZ
770 PRINT:PRINT"●●●":PRINTTAB(32);I
780 W2$=STR$(I):POKEH3,0
790 PRINT"SQQQQQQQQQQQQQQQQ":FORZZ=1TO220:PRINT" ";:NEXTZZ:PRINT
800 PRINT"●●●●●BITTE RUHEN SIE SICH WIEDER 1Q"
810 PRINT"MINUTE AUS ! "
820 YY=15:A1=TI
830 GOSUB1090:GOSUB500
840 IF(TI-A1)/60<60 THEN830
850 POKEH3,16:POKEH2,200:POKEH1,15
860 FORI1=1TO1500:NEXTI1
870 POKEH3,0
880 PRINT:PRINT"●●●":PRINTTAB(32);I
890 FORLL=1TO400:NEXTLL
900 PRINT"SQQQQQQQQQQQQQQQQ":FORZZ=1TO200:PRINT" ";:NEXT
910 FORA1=1TO1000:NEXTA1
920 W3$=STR$(I)
930 PRINT"♥QQBITTE LEGEN SIE EINE DATENKASSETTEQ"
940 PRINT"EIN UND DRUECKEN SIE EINE TASTE !QQ"
950 GETT$:IFT$=""THEN950
960 POKE243,122:POKE244,2
970 OPEN1,1,1
980 PRINT#1,N$
990 PRINT#1,D$
1000 PRINT#1,W1$
1010 PRINT#1,W2$
1020 PRINT#1,W3$
1030 PRINTN$,D$,W1$;W2$;W3$
1040 CLOSE1
1050 END
1060 GOSUB1090
1070 YY=3:GOSUB500
1080 GOTO1060
1090 SYS826:W=PEEK(1000)+256*PEEK(1001)+65536*PEEK(1002)
1092 I=INT(3356400/W)
1095 IFI<400RI>200THEN1090
1100 RETURN
1110 PRINTI:GOTO1090
1120 RETURN
1130 PRINT"♥QQ   ABGESPEICHERTE WERTEQQ"
1140 POKE243,122:POKE244,2
1150 OPEN1
1160 INPUT#1,N$,D$,W1$,W2$,W3$
1170 PRINTN$,D$,W1$;TAB(25);W2$;TAB(30);W3$
1180 CLOSE1
1190 GETT$:IFT$=""THEN1190
1200 RUN1140
```

Bild 4. Das für den Mikrocomputer PET-2001 geschriebene Basic-Programm wurde nach medizinischen Gesichtspunkten entwickelt und generiert auch Töne

tes Platinen-Layout und *Bild 3* den dazugehörigen Bestückungsplan. Die am Ausgang befindliche Leuchtdiode blinkt bei richtiger Einstellung im Rhythmus des Blutdurchflusses.

Das Programm in *Bild 4* wurde nach medizinischen Gesichtspunkten entwickelt und für den PET-2001 geschrieben. Die Zeilen 330...360 generieren ein Maschinenprogramm für die 6502-CPU, ein Umschreiben für andere Computer ist also leider mit einigem Aufwand und Denkarbeit verbunden. Das Basic-Programm ist weitgehend selbsterklärend; der Pulssensor ist am Pin PA 7 des User-Port (Bit 8) anzuschließen. Ein an CB 2 angeschlossener Nf-Verstärker dient zur Wiedergabe der generierten Töne.

Willi Billen
Willi Gietmann

Messen, Steuern, Regeln

Glühlampe, Schrittmotor und Schalter sind die anschaulichen Elemente eines neuen Lehrmodells, mit dem Siemens die Möglichkeiten der Mikrocomputer demonstrieren will. Messen, Steuern und Regeln sind exemplarisch dargestellt: Der Widerstand der Glühlampe wird nach einer vorgehbaren Einschaltdauer gemessen und angezeigt; den Schrittmotor steuert ein ebenfalls vorgebbares Programm mit bis zu acht Wertepaaren (Schrittzahl und Drehrichtung); der Schalter schließlich „simuliert" die Störgröße einer Regelstrecke dritter Ordnung.

Das Lehrmodell basiert auf dem SMP-Mikrocomputersystem für Industrieanwendungen aus Standardbaugruppen. Die Applikationsschrift „Demonstrationsmodell für die Anwendungen Messen, Steuern und Regeln" kann unter der Bestellnummer B/2191 kostenlos bezogen werden von Siemens AG, ZVW 85, Postfach 1500, D-8510 Fürth 2. Das Format ist DIN A4.

Rudolf Hofer

BASIC vor dem Handbuch

1. Teil

Heimcomputer sind heute so preiswert, daß sie innerhalb der finanziellen Möglichkeiten eines großen Anwenderkreises liegen. Häufig übersieht der Käufer aber, daß Computer Werkzeuge sind, die nur dann ihre Leistung preisgeben, wenn man sie zu handhaben versteht. Die mitgelieferten Handbücher klären zwar über spezifische Eigenheiten einer Maschine auf, die wahren Probleme des Anfängers jedoch können und sollen sie nicht behandeln. Was der Anfänger braucht, ist eine behutsame Heranführung an den Computer – ein allmähliches Vertrautwerden mit dem, was die Maschine versteht und was sie störrisch ablehnt. Dieses Gefühl für den Umgang mit dem Rechner zu vermitteln, ist das Ziel dieses Beitrages. Er soll Hand- und Lehrbücher nicht ersetzen, sondern ergänzen.

Obwohl sich die folgenden Angaben auf den TRS-80 beziehen, beschränkt sich unsere Serie keineswegs auf diesen Typ. Alle Befehle und Beispiele wurden so gewählt, daß sie allgemein gültig sind. Besonderer Wert wurde darauf gelegt, daß nicht allein die rechnerischen Fähigkeiten des Computers offenbar werden. Mathematische Problemstellungen, mit Ausnahme der Grundrechenarten, fehlen daher völlig.

Organe des Computers

In der Grundausstattung hat unser Computer zwei Möglichkeiten, Daten aufzunehmen (Tastatur, Kassette), und zwei, Daten auszugeben (Bildschirm, Kassette). Unmittelbar vom Menschen nimmt er nur Eingaben über die Tastatur an. Grundsätzlich unterscheiden wir zwei Arten von Eingaben: Bei der ersten handelt es sich um Anweisungen, die unmittelbar ausgeführt werden, sogenannte Kommandos; bei der zweiten werden Befehle für ein Programm eingetippt, das erst zu einem späteren Zeitpunkt ablaufen soll. Der Computer zeigt an, daß er für Kommando- und Befehlseingabe bereit ist, indem er ein immer gleichbleibendes Antwortzeichen (z.B. > READY) auf den Schirm schreibt.
Ist das Programm fertig, kann es mit Hilfe des Kommandos RUN gestartet werden.
An dieser Stelle ein Hinweis, den man sich für die erste Zeit einbleuen sollte:
Nach jeder Eingabe (also nach einem Kommando wie RUN oder einer fertig eingetippten BASIC-Zeile) ist die Enter-Taste zu drücken (bei manchen Modellen Return), erst dann fühlt sich der Computer angesprochen.

Ein kleines Programm

Schreiben wir zunächst ein Programm, das aus zwei Zeilen besteht:

```
100 PRINT "VERSUCH"
110 GOTO 100
```

Wenn Sie sich vertippt haben, können Sie mit Linkspfeil (bei manchen Modellen BS oder DEL) je ein Zeichen zurückgehen. Nach Enter ist dieser Versuch allerdings zwecklos. In diesem Fall schreiben Sie die Zeile einfach neu (später entnehmen wir dem Handbuch unter dem Stichwort „Editieren" eine bequemere Möglichkeit zum Korrigieren). Die alte Zeile wird dann überschrieben. Um eine Zeile nur zu löschen, tippt man die Zeilennummer ein und dann Enter.

Das Programm steht jetzt im Arbeitsspeicher. Auch wenn der Bildschirm gelöscht wird (z.B. mit der CLR-Taste), wird es nicht zerstört. Zum Beweis geben wir das Kommando LIST ein, und das Programm erscheint wieder auf dem Bildschirm. Bevor wir es starten, wollen wir überlegen, was passieren wird und warum. Die Nummern vor den eigentlichen Befehlen nennen wir Zeilennummern:

Jede BASIC-Zeile erhält eine Zeilennummer!

Sie wird willkürlich gewählt zwischen 1 und 65529 (verschiedene Modelle erlauben maximal 10000). Beginnend bei der niedrigsten Zeilennummer arbeitet der Computer alle Befehle in aufsteigender Reihenfolge ab, bis er auf einen Sprungbefehl trifft, der ihm sagt, wo er weitermachen soll.

Um später Befehle einfügen zu können, läßt man im allgemeinen einen Abstand von 10 zwischen den Zeilennummern! Der erste Befehl in unserem Programm lautet

```
PRINT
```

Er besagt: Drucke alles, was in den nachfolgenden Anführungszeichen eingeschlossen ist. Der Computer druckt also VERSUCH und geht zum nächsten Befehl. Der lautet:

```
GOTO 100
```

Also geht der Computer zurück zu Zeile 100 und findet wieder den PRINT-Befehl. Das Spiel setzt sich endlos fort.
Starten wir das Programm jetzt mit

```
RUN
```

Danach muß natürlich die Enter-Taste gedrückt werden! Das war jetzt der letzte Hinweis darauf, in Zukunft müssen Sie selber dran denken.
Falls sie keine Fehler gemacht haben, sollte auf dem Bildschirm jetzt pausenlos VERSUCH untereinander gedruckt werden.

Da das Programm kein natürliches Ende findet, brechen wir es gewaltsam mit der Break-Taste ab – manche Modelle verlangen hier CTRL C (C bei gleichzeitig betätiger CTRL-Taste drücken). Das neuerliche LIST-Kommando beweist uns, daß es der Computer immer noch nicht vergessen hat.

Erweitern wir die Zeile 100 jetzt um einen schlichten Strichpunkt am Ende:
100 PRINT „VERSUCH";
Wenn das Programm jetzt gestartet wird, druckt der Computer das Wort Versuch fortlaufend, ohne Absatz und ohne Abstand. Daraus folgern wir:
Steht ein Strichpunkt nach dem PRINT-Befehl, wird der nächste Ausdruck unmittelbar an den vorhergehenden angeschlossen!
Um den Computer für ein neues Programm vorzubereiten, gibt man das Kommando NEW. Das alte wird dann vollständig gelöscht.
Soll das alte Programm nicht gelöscht werden, schließt man das neue einfach an, indem man höhere Zeilennummern verwendet. Mit RUN 200 kann dann beispielsweise ein Programm ab Zeile 200 gestartet werden. In diesem Fall ist es sinnvoll, ein eigenständiges Programm mit dem Befehl END abzuschließen, da der Computer sonst unter Umständen versehentlich in ein weiteres Programm „hineinläuft".

Ein bißchen Rechnen

Computer bezeichnet man oft auch als Rechner. – Nicht ohne Grund, denn eine ihrer wesentlichen Eigenschaften ist, daß sie rechnen können. Allerdings sind sie nicht nur dazu imstande – doch davon soll später die Rede sein. Vorerst wollen wir uns ihren fundamentalen mathematischen Möglichkeiten zuwenden.
Ein Beispiel:
```
100 A=1
110 B=10
120 C=A+B
130 PRINT C
```
In den ersten beiden Zeilen finden wir sogenannte Zuweisungen. Den Variablen A und B wird je ein Wert zugewiesen. Variable werden (vorerst) mit einem Buchstaben gekennzeichnet. Sie heißen deshalb so, weil sie innerhalb eines Programms verschiedene Werte annehmen können. Die Werte werden ihnen mit einem „="-Zeichen zugewiesen. Dabei gilt immer:
Die Variable links vom „="-Zeichen erhält den Wert rechts vom „="-Zeichen!
Nun kann auf der rechten Seite aber auch ein sogenannter Ausdruck stehen, wie in Zeile 120.
Ein Ausdruck besteht aus Variablen und Zahlenwerten, die miteinander verknüpft sind!
Die im Ausdruck auftauchenden Variablen müssen vorher bereits auf einen bestimmten Wert gesetzt (definiert) worden sein, andernfalls erhalten sie den Wert Null.

Betrachten wir noch Zeile 130. Im PRINT-Befehl steht jetzt kein Text in Anführungszeichen, sondern eine Variable. In diesem Fall wird der Wert, den sie an dieser Stelle gerade hat, ausgedruckt.
Man kann aber auch beide PRINT-Typen miteinander verbinden. Ändern wir die Zeile 130 folgendermaßen:
130 PRINT „C HAT DEN WERT"; C
Wie erwartet, wird der Text innerhalb der Anführungszeichen unverändert ausgedruckt, während dahinter ohne Absatz (wegen des Strichpunktes) der augenblickliche Wert von C erscheint. Vor der Variablen wird automatisch ein Leerzeichen eingefügt. Eine weitere Möglichkeit besteht darin, den Ausdruck direkt im PRINT-Befehl berechnen zu lassen. Unser Beispiel würde dann etwa folgendermaßen aussehen:
```
100 A=1
110 B=10
120 PRINT "A+B=";A+B
```

Die Grundrechenarten

Die vier Grundrechenarten werden in BASIC durch folgende Zeichen dargestellt:
Addition +
Subtraktion –
Multiplikation *
Division /
Jeder Ausdruck wird von links nach rechts gearbeitet. Allerdings werden Division und Multiplikation vor Addition und Subtraktion ausgeführt. Klammern werden von innen nach außen abgearbeitet. In Zweifelsfällen empfiehlt es sich, von Klammern Gebrauch zu machen. Jede geöffnete Klammer muß auch wieder geschlossen werden.

Um die nötige Sicherheit in der Umstzung von arithmetischen Ausdrücken in BASIC-Anweisungen zu erhalten, sollte man verschiedene Beispiele durchspielen. Den Computer kann man zu diesem Zweck im sogenannten Direkt-Modus betreiben. Man tippt dabei PRINT ohne Zeilennummer ein, gefolgt von dem zu berechnenden Ausdruck. Das Ergebnis wird nach dem Betätigen der Enter-Taste unmittelbar ausgegeben.

Beispiele	Ergebnis
PRINT 3–4*8	–29
PRINT 4*(3+5)–2	30
PRINT 8–3*6/3	2
PRINT 2*3.14/(4*4.5)	0.348889
PRINT 2*3.14/4*4.5	7.065

usw.
Zwei Dinge haben uns diese Beispiele zusätzlich gezeigt: Der Computer ist auch wie ein normaler Taschenrechner anzuwenden!
Statt des „Dezimalkommas" wird in BASIC ein Punkt verwendet!

Vorsicht ist bei der Multiplikation geboten. In der üblichen arithmetischen Schreibweise verzichtet man oft auf ein besonderes Multiplikationszeichen (z.B. ist 2R gleichbedeutend mit 2*R). In BASIC ist das nicht erlaubt.
Jede Multiplikation wird durch das Zeichen „" ausgedrückt!*

Eine konkrete Aufgabe

Eine der Hauptschwierigkeiten des Anfängers besteht darin, zu begreifen, wofür so ein Computer gut ist. Nun hat es wenig Sinn, das mit breiten Worten zu erklären. Es fehlt im Augenblick einfach noch am Gespür für die Arbeitsweise, die Kapazität und die Geschwindigkeit der Maschine. Aber dieses Gespür wird sich sehr bald einstellen. Eine Weile müssen wir uns aber noch gedulden, denn die nächsten Beispiele lassen noch nichts von der Leistungsfähigkeit eines Computers erkennen. Trotzdem wollen wir versuchen, von Anfang an nützliche Anwendungen zu finden. Die Aufgabe lautet:

Aus einem Betrag, den der Benutzer eingibt, soll die Mehrwertsteuer berechnet werden. Betrag, Steuer und Summe sind auszudrucken. Danach soll der Computer für eine neue Eingabe bereit sein. (Festlegung: B=Betrag, M=Mehrwertsteuer.)

Bild 1a zeigt ein sogenanntes Flußdiagramm, das den Programmablauf vorgibt. Nach dem Start (entspricht dem RUN-Kommando) soll der Computer auf eine Eingabe vom Benutzer warten. Wie wir gleich sehen werden, ist dafür ein neuer Befehl nötig. Der Betrag wird der Variablen B zugewiesen und beim nächsten Schritt mit 0,13 multipliziert – das entspricht 13 % vom Betrag. Dieser neue Wert soll der Variablen M zugewiesen werden. Schließlich sind B und M sowie die Summe B+M auszudrucken. Nun folgt ein Sprung

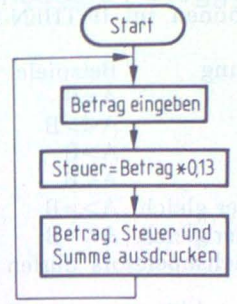


Bild 1a. Flußdiagramm zur Berechnung der Mehrwertsteuer

```
90 PRINT "BETRAG";
100 INPUT B
110 M=B*0.13
120 PRINT "BETRAG: ";B;" DM   MWST.: ";M;" DM   SUMME: ";B+M;" DM"
130 GOTO 90
```

Bild 1b. Programm zur Berechnung der Mehrwertsteuer

Beispiel:

```
BETRAG? 100
BETRAG:  100  DM   MWST.:  13  DM   SUMME:  113  DM
BETRAG? 50
BETRAG:  50  DM   MWST.:  6.5  DM   SUMME:  56.5  DM
BETRAG? 21
BETRAG:  21  DM   MWST.:  2.73  DM   SUMME:  23.73  DM
BETRAG?
```

◄ Bild 1c. Beispiele zum Programm in Bild 1b

zum Programmanfang. Ein neuer Betrag wird vom Benutzer erwartet.

Das Programm stellt wieder eine endlose Schleife dar und kann nur mit der Break-Taste unterbrochen werden. Es sieht folgendermaßen aus:

```
100 INPUT B
110 M=B*0.13
120 PRINT B,M,B+M
130 GOTO 100
```

Neu ist der Befehl INPUT.

Trifft das Programm auf einen INPUT-Befehl, dann hält es an, druckt ein Fragezeichen und wartet auf eine Eingabe von der Tastatur, die der nach INPUT stehenden Variablen zugewiesen wird!

Werden mehrere Variablen nach INPUT durch Kommas getrennt (z.B. INPUT A,B,C), dann erwartet der Computer entsprechend viele Eingaben.

In Zeile 110 finden wir eine einfache Zuweisung, die uns schon bekannt ist. B wird hier nicht verändert, da es auf der rechten Seite steht.

In Zeile 120 werden die auszudruckenden Variablen durch Komma getrennt. Dies bewirkt, daß der folgende Ausdruck an der nächsten Tabulator-Position fortgesetzt wird. Tabulator-Positionen befinden sich an jeder sechzehnten Stelle. Auch Texte können mit Komma getrennt werden. Nützlich ist das Komma im PRINT-Befehl, wenn man ein einheitliches Format erreichen will, z.B. beim Ausdruck von Tabellen. In Zeile 130 schließlich finden wir den schon bekannten Sprungbefehl.

Wenn wir das Programm jetzt laufen lassen, erscheint zunächst ein Fragezeichen auf dem Bildschirm. Jetzt geben wir einen beliebigen Betrag ein und schließen mit Enter ab. Der Computer druckt die drei Zahlen aus und erwartet sofort wieder einen neuen Betrag.

Zwar tut er genau das, was wir von ihm verlangt haben, doch die Zahlen werden ziemlich beziehungslos ausgegeben. Mit Hilfe von eingefügten Texten wollen wir jetzt Klarheit in das Programm bringen.

Doch dieses Problem sollen Sie selbst lösen. Folgende Forderungen wollen wir stellen: Bevor er die Eingabe des Betrages verlangt, soll der Computer ausdrucken: BETRAG? (Hinweis: Das Fragezeichen wird vom INPUT-Befehl ausgegeben). Bei der Ausgabe soll ein Kommentar nach folgendem Muster mit ausgedruckt werden:
BETRAG: 100 DM MWST.: 13 DM SUMME: 113 DM
Die Zahlen sind hier willkürlich gewählt und entsprechen den Variablen A und B bzw. dem Ausdruck A+B.
Die Lösung finden Sie in *Bild 1b*; den Programmablauf zeigt *Bild 1c*.

Der Computer trifft eine Entscheidung

Die bisherigen Programme waren alle endlose Schleifen, die man gewaltsam unterbrechen mußte. Nun wollen wir dem Computer eine Aufgabe geben, die er von selbst abschließt. Bleiben wir bei der Mehrwertsteuer:

Das Programm soll eine Tabelle ausdrucken, in deren linker Spalte Beträge von 1...100 stehen. In der rechten Spalte steht die zugehörige Mehrwertsteuer. Die Tabellenüberschrift lautet dementsprechend
BETRAG MWST.
Ist der Ausdruck beendet, soll der Computer zum Beweis ENDE drucken.

Die Lösung anhand eines Flußdiagramms zeigt *Bild 2*. Das gibt uns die Möglichkeit zu ein paar Bemerkungen über Flußdiagramme im allgemeinen: Wir sehen, daß die ovalen Gebilde mit dem eigentlichen Programm nichts zu tun haben, sie markieren in diesem Fall nur Anfang und Ende. Man benutzt sie auch, um den Übergang zu einem anderen Programmteil darzustellen, der in derselben Zeichnung keinen Platz mehr gefunden hat. Die rechteckigen Kästen stellen einzelne Befehle oder ganze Befehlsfolgen dar. Es bleibt jedem selbst überlassen, wieviel er da hineinpacken möchte. Oft geht man von einem groben Flußdiagramm aus und verfeinert es dann nach und nach, bis man die BASIC-Befehle gewissermaßen schon herauslesen kann. Die Rauten stellen Abfragen nach einer Bedingung dar. Ist die Bedingung erfüllt, macht das Programm am Ausgang „ja" weiter, ist sie nicht erfüllt, macht es bei „nein" weiter. Im Beispiel wird einfach danach gefragt, ob die Variable B schon größer als 100 ist.

Will man korrekt sein, dann muß man jede Verbindung zwischen den Elementen des Flußdiagramms mit einem Pfeil versehen, der die Richtung des Programmflusses angibt. Leider wird das sehr häufig vergessen. Zurück zum Beispiel:

Nachdem die Tabellenüberschrift gedruckt ist, wird dem Betrag B zunächst der Wert 1 zugewiesen; unsere Tabelle soll ja mit 1 beginnen. Danach werden B und B*0.13 (entspricht wieder der MwSt.) ausgedruckt. B wird jetzt um 1 erhöht. Hier sehen wir, daß eine Zuweisung nicht mit einer mathematischen Gleichung verwechselt werden darf: Die Variable auf der linken Seite erhält lediglich den Wert, der sich aus dem Ausdruck auf der rechten Seite ergibt. Daß dieselbe Variable auch in diesem Ausdruck vorkommt, spielt überhaupt keine Rolle. Beim ersten Durchlauf hat B also den Wert 1. Durch die Zuweisung B=B+1 wird zu diesem Wert eine 1 addiert. Das Egebnis wird B zugewiesen. Wir sehen, daß B der Reihe nach die Werte 1, 2, 3...101 annimmt. Danach wird das Programm beim Ausgang „ja" der Abfrage-Raute fortgesetzt, d.h. es wird ENDE ausgegeben.

Wenn kein weiterer Befehl mehr folgt, meldet sich der Computer mit READY und zeigt damit an, daß er wieder aufnahmebereit ist. Manche BASIC-Versionen verlangen als letzten Befehl END. Die meisten Heimcomputer können aber darauf verzichten.

Bild 3 zeigt die Realisierung des Programms. Neu ist der Befehl in Zeile 140, der die Abfrage der Bedingung B≦100 darstellt.

IF B <=100 THEN 120 bedeutet,

wenn B kleiner oder gleich 100 ist, dann wird das Programm bei Zeile 120 fortgesetzt, sonst bei der nächsten Zeilennummer.

Wie wir sehen, muß die Abfrage im Flußdiagramm nicht mit der im BASIC-Programm identisch sein. Im IF-THEN-Befehl wird der Sprung immer nur dann ausgeführt, wenn die Bedingung erfüllt ist. Wir müssen sie deshalb entsprechend wählen. Praktisch ist das allerdings kein Problem, da es zu jeder Bedingung eine Umkehrung gibt.

Um sicherzugehen, daß das Programm richtig ist, spielt man es am besten in Gedanken durch, bevor man es laufen läßt. In unserem Fall würden wir etwa folgende Überlegungen anstellen:

Die Tabellenüberschrift darf nur einmal ausgedruckt werden, also darf das nicht innerhalb der Schleife geschehen. BETRAG und MWST. werden durch Komma getrennt, dadurch beginnt der Ausdruck bei Position 0 und wird bei Position 16 fortgesetzt. Die auszudruckenden Variablen müssen später ebenfalls durch Komma getrennt werden, damit das Format stimmt.

Vor dem Eintritt in die Schleife wird der Variablen B, die den Betrag repräsentieren soll, der erste interessierende Wert (also 1) zugewiesen. Nach dem Ausdrucken der Variablen wird B erhöht. Da B jetzt erst abgefragt wird, muß sichergestellt sein, daß beim Wert 100 noch einmal zurückgesprungen wird und B bzw. B*0.13 ausgedruckt werden. Es soll also immer zurückgesprungen werden, wenn B kleiner oder gleich (ausgedrückt durch <=) 100 ist. Beim nächsten Durchlauf wird für B der Wert 100 ausgegeben. Danach wird B erhöht, und die Sprungbedienung ist nicht mehr erfüllt.

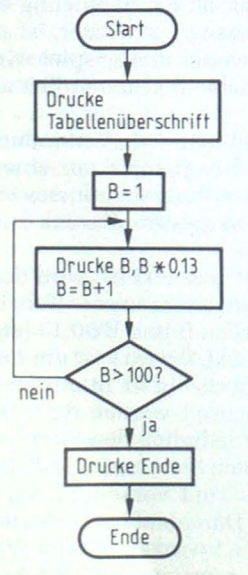


Bild 2. Flußdiagramm zum Erstellen einer Mehrwertsteuertabelle

```
100 PRINT "BETRAG","MWST."
110 B=1
120 PRINT B,B*.13
130 B=B+1
140 IF B<=100 THEN 120
150 PRINT "ENDE"
```

```
BETRAG          MWST.
 1               .13
 2               .26
 3               .39
 4               .52
 5               .65
```

Bild 3. Programm zum Erstellen einer Mehrwertsteuertabelle

```
100 PRINT "BETRAG","MWST."
110 FOR B=1 TO 100
120 PRINT B,B*.13
130 NEXT B
140 PRINT "ENDE"
```

Bild 4. Dieses Programm entspricht dem in Bild 3

Folglich wird der nächste Befehl (PRINT "ENDE") ausgeführt. Folgende Vergleiche können im IF-THEN-Befehl durchgeführt werden:

Vergleichsoperator	Bezeichnung	Beispiele
=	gleich	A=B
<>	ungleich	A<>B
>	größer	A>B
<	kleiner	A<B
>=	größer oder gleich	A>=B
<=	kleiner oder gleich	A<=B

Auf beiden Seiten des Vergleichsoperators dürfen dabei auch Ausdrücke stehen, z.B.

IF 2*P/4=A+B+C THEN ...

Wir haben jetzt einen der wichtigsten Befehle kennengelernt. Zum erstenmal hat der Computer eine Entscheidung getroffen. Dennoch läßt sich das letzte Beispiel mit Hilfe eines weiteren Befehls eleganter lösen.

Der Computer dreht Schleifen

Die Alternativlösung zum Programm in Bild 3 ist in *Bild 4* dargestellt. Betrachten wir die Zeilen 110 und 130 – sie gehören eng zusammen, und eine kann ohne die andere nicht auftreten. Diese beiden Zeilen stellen einen einzigen Befehl dar: den FOR-NEXT-Befehl. Er besagt in unserem Beispiel: Weise B der Reihe nach die Werte 1, 2, 3...100 zu und führe dazwischen den oder die Befehle aus, die von der FOR- und der NEXT-Zeile eingeschlossen werden. Ist der Wert 100 erreicht, wird der nächste Befehl (nach NEXT) im Programm ausgeführt. B ist in diesem Fall die Laufvariable, weil sie eine Reihe von Werten bis zu einem Endwert durchläuft. Die Schrittweite ist der Wert, um den B nach jedem Durchlauf verändert wird (im Beispiel also 1). Sie kann auch „krumme" Werte annehmen:

FOR B=0 TO 100 STEP 0.25

würde z.B. zur Folge haben, daß B bei jedem Durchlauf um 0,25 erhöht wird. Die Schrittweite ist 1, wenn STEP... fehlt.

Programme auf Kassette aufbewahren

Wir sind jetzt in der Lage, kleinere Programme selbst zu schreiben. Mit unseren Kenntnissen wird auch die Länge der Programme zunehmen. Um sie nicht jedesmal neu eintippen zu müssen, speichern wir sie auf Kassette ab und laden sie bei Bedarf wieder in den Computer. Das geht sehr einfach mit Hilfe der Kommandos CSAVE und CLOAD. Manche Heimcomputer verlangen hier andere Kommandos, die aber auf die gleiche Weise benutzt werden (siehe Handbuch).

Das Abspeichern geht so vor sich:

Verbindung zwischen Computer und Kassettenrecorder herstellen.

Leerkassette einlegen.

Recorder auf Aufnahme (das Band läuft noch nicht los, weil der Computer über ein Relais den Motorstrom unterbrochen hat).

CSAVE "A" eingeben (anstelle von A kann jeder beliebige Buchstabe als Kennung für das Programm gewählt werden).

Jetzt wird der Motorstrom vom Computer „freigegeben" und das Programm aufgenommen. Das Band stoppt automatisch, wenn der Aufnahmevorgang beendet ist.

Wie der Recorder normal bedient werden kann (z. B. Zurückspulen), steht im Handbuch. Wir erinnern uns, daß der Motorstrom ja unterbrochen ist. Herausziehen des Verbindungskabels zum Computer (aus dem Kassettenrecorder!) hilft aber immer.

(Fortsetzung folgt)

Hans-Joachim Regge

Video für den AIM-65

Der mancherorts schon zu einem Renner gewordene AIM-65, der auch als professionelles Datenerfassungsgerät eingesetzt wird, steht mitunter bei Vorführungen im Schatten von Geräten, die (leider oft zu spät erkannt) bis auf das äußerste abgemagert sind, aber lustige Spiele auf einem Bildschirm darstellen können. Für manche Zwecke ist nun ein Video-Anschluß wirklich vorteilhafter.

Das Programm im *Bild* ermöglicht die Verwendung einer einfachen Videoplatine, z. B. aus FUNKSCHAU 1978, Heft 24, oder Solid State Music VB1B. Ein Controller ist nicht erforderlich, die Platine muß lediglich einen direkten Speicherzugriff ermöglichen. Cursor-Steuerung, Scrolling und alle anderen Funktionen werden per Software erledigt, Zeitschleifen werden nicht benötigt. Hierdurch ist diese Ansteuerungsmethode anderen sowohl in der Geschwindigkeit als auch in der Flexibilität überlegen – dafür werden für das Video-Betriebsprogramm 512 Bytes belegt. Es wird aber auch etwas geboten: Video on/off per Software, Clear Screen, Cursor Home, Up, Down, Left und Right, 128 Schriftzeichen weiß auf schwarz oder schwarz auf weiß (auch gemischt), sichtbarer, nicht „zerstörender" Cursor sowie natürlich autom. Scrolling. Cursor-Steuerungszeichen sowie Umschaltzeichen Groß/Kleinschreibung (oder soll's Griechisch sein?) können in Texten mit eingebaut sein, und wer eine feststehende Überschrift oder ein „Fenster" wünscht... auch diese Dinge lassen sich einbauen.

In der Zero-Page werden D6...DE belegt (für Maschinenprogramme und Assemblerarbeit geeignet), das Programm liegt bei 0E00...0FFF und das Video-RAM bei 1000...13FF. Beim Assemblieren von und auf Band muß Video mit Control/F3 vorher ausgeschaltet, danach mit F3 wieder eingeschaltet werden. Für Basic müssen die benutzten Zero-Page-Adressen oberhalb von F0 liegen; das erfordert eine Programmänderung, bei der man sinnvollerweise das Zero-Page-Referenzprogramm aus FUNKSCHAU 1980, Heft 6, Seite 108, benutzt (FROM = 0E00, TO = 0FFF).

Die Bedienung

Zunächst lädt man das Programm und drückt – bei angeschlossener Video-Platine – die Taste F3. Der Bildschirm zeigt jetzt „ON" und „ROCKWELL AIM 65" bzw. „SIEMENS PC 100". Jetzt befindet man sich im Monitorprogramm, wobei alle gewünschten Funktionen vorhanden sind. Die Videoausgabe bleibt auch nach einem Reset wirksam, da der DILINK-Vektor (A406/A407) vom Programm verändert wurde. Jedes auf dem AIM-Display abgebildete Zeichen erscheint auch auf dem Video-Display. Auch DEL wird auf dem Bildschirm nachgebildet, bis das Display nach links rollt.

Drückt man die Taste CTRL zusammen mit Buchstabentasten, so werden Kontrollzeichen erzeugt, die den Cursor steuern. Diese Zeichen werden vom AIM weder verwendet noch abgebildet, dürfen aber in einem anzuzeigenden Text vorhanden sein, um Steuerfunktionen auszulösen. Die *Tabelle* zeigt die Wirkung der CTRL-Funktionen.

CTRL	F3	: Video off (betätigen vor dem Assemblieren vom und zum Band), wird oben links angezeigt
CTRL	Q	: Cursor Home (links oben)
CTRL	X	: Bildschirm löschen, Cursor Home
CTRL	S	: Kleinbuchstaben nachfolgend schreiben
CTRL	A	: Nachfolgend wieder Großbuchstaben schreiben
CTRL	E	: Griech. Buchstaben/Sonderzeichen nachfolgend
CTRL	D	: Cursor 1 Zeile tiefer
CTRL	U	: Cursor 1 Zeile höher
CTRL	L	: Cursor 1 Stelle nach links
CTRL	R	: Cursor 1 Stelle nach rechts
CTRL	W	: reverse (schwarz auf weiß) nachfolgend schreiben
CTRL	W nochmals	: Wieder weiß auf schwarz schreiben

```
0112 4C 00 0E 09 0E54 48 A4 D8 B1 0EAC 09 20 4C B8 0F04 A5 DA 85 DC 0F5C 20 00 0F 60 0FB4 40 85 DB A5
0E00 A9 4A 8D 06 0E58 DB 49 80 91 0EB0 0E 68 29 BF 0F08 A5 D7 85 D6 0F60 C9 18 D0 1B 0FB8 DC E9 00 85
0E04 A4 A9 0E 8D 0E5C DB A5 D6 C9 0EB4 4C B8 0E 68 0F0C 18 69 40 AA 0F64 20 4B 0F A0 0FBC DC 60 C9 0C
0E08 07 A4 6E 11 0E60 17 D0 0F A9 0EB8 A4 D8 05 DD 0F10 C6 D6 CA 8A 0F68 00 A9 20 91 0FC0 D0 19 A5 DC
0E0C A4 A9 00 85 0E64 00 C5 DD F0 0EBC 91 DB C8 C4 0F14 A8 B1 DB A4 0F6C DB C8 C4 D7 0FC4 C9 13 D0 06
0E10 D8 85 DD 85 0E68 05 85 DD 4C 0EC0 D7 F0 03 84 0F18 D6 91 DB D0 0F70 D0 F7 A0 00 0FC8 A5 DB C9 C0
0E14 DE 85 D9 A9 0E6C 72 0E A9 80 0EC4 D8 60 A0 00 0F1C F3 20 DC 0F 0F74 20 DC 0F 90 0FCC F0 BB A5 DB
0E18 20 85 D7 A9 0E70 85 DD A5 D6 0EC8 84 D8 4C FB 0F20 90 E6 A4 D7 0F78 F0 84 D8 20 0FD0 18 69 40 85
0E1C 10 85 DA 20 0E74 20 87 0E A4 0ECC 0E 98 A9 20 0F24 A9 20 88 91 0F7C 4B 0F 60 C9 0FD4 DB A5 DC 69
0E20 64 0F 20 4B 0E78 D8 B1 DB 49 0ED0 A4 D8 F0 09 0F28 DB D0 FB 60 0F80 0C D0 0F A4 0FD8 00 85 DC 60
0E24 0F 20 3B E8 0E7C 80 91 DB 68 0ED4 C0 15 B0 05 0F2C C9 01 D0 05 0F84 D8 F0 03 C6 0FDC 18 A5 DC C9
0E28 20 FA E6 4C 0E80 A8 68 AA 68 0ED8 88 91 DB 84 0F30 A9 00 85 DE 0F88 D8 60 A4 D7 0FE0 13 90 08 A5
0E2C BF E0 20 4B 0E84 4C 05 EF C9 0EDC D8 60 C9 0D 0F34 60 C9 05 D0 0F8C 88 84 D8 4C 0FE4 DB C9 C0 90
0E30 0F 20 3B E8 0E88 8D F0 5A C9 0EE0 D0 15 4C E5 0F38 05 A9 02 85 0F90 A4 0F C9 12 0FE8 02 38 60 A5
0E34 20 F1 E6 20 0E8C 20 90 4F A8 0EE4 0E A0 00 84 0F3C DE 60 C9 13 0F94 D0 0A A4 D8 0FEC DB 69 40 85
0E38 3B E8 20 13 0E90 BA BD 07 01 0EE8 D8 08 48 20 0F40 D0 05 A9 01 0F98 C8 C4 D7 B0 0FF0 DB 90 03 E6
0E3C EA A9 05 8D 0E94 C9 E7 D0 07 0EEC 9E EB 20 FB 0F44 85 DE 60 C9 0F9C EC 84 D8 60 0FF4 DC 18 60 BD
0E40 06 A4 A9 EF 0E98 BD 06 01 C9 0EF0 0E 20 AC EB 0F48 11 D0 0D A0 0FA0 C9 15 D0 1A 0FF8 44 00 C9 02
0E44 8D 07 A4 4C 0E9C F2 F0 2E 98 0EF4 68 28 60 C9 0F4C 00 A5 D9 85 0FA4 A5 DC C5 DA 0FFC 60 4C 03 B0
0E48 BF E0 C9 1E 0EA0 29 7F 48 A5 0EF8 04 D0 31 20 0F50 DB A5 DA 85 0FA8 D0 06 A5 DB
0E4C F0 E0 85 D6 0EA4 DE F0 10 C9 0EFC DC 0F 90 C5 0F54 DC 84 D8 60 0FAC C5 D9 F0 D9
0E50 48 8A 48 98 0EA8 01 D0 06 68 0F00 A5 D9 85 DB 0F58 C9 03 D0 04 0FB0 A5 DB 38 E9
```

Zusammen mit einer einfachen DMA-Video-Platine ohne Controller, z. B. aus FUNKSCHAU 1978, Heft 24, ermöglicht dieses Programm eine Videoausgabe bei AIM-65 und PC-100

Von TI verschmäht:

Der DAI-Computer

Der von DAI hergestellte Personal Computer (*Bild*) wurde ursprünglich im Auftrag von Texas Instruments entwickelt. Er sollte „der" TI-Heimcomputer werden. Als sich TI schließlich aber doch entschloß, lieber selbst einen Heimcomputer zu entwickeln und die eigene CPU TMS 9900 zu verwenden, vertrieb DAI das Gerät selbst.

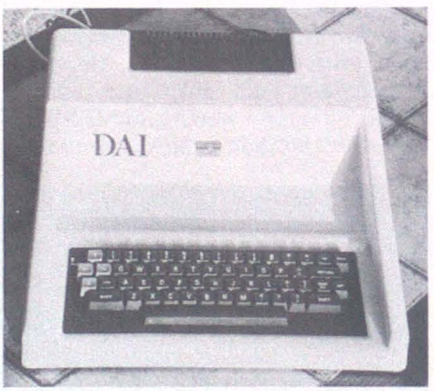
DAI ist eine belgische Firma, die praktisch zu den Pionieren der Mikrocomputertechnik gehört. Seit Jahren stellt das Unternehmen professionelle Europakarten-Computersysteme her. Erst der Auftrag von Texas Instruments bewegte DAI zum Einstieg in den Personal-Computer-Markt.

Verständlicherweise findet man beim DAI-Computer, der seit Mitte des Jahres erhältlich ist, einige Gemeinsamkeiten mit dem von TI herausgebrachten TI-99/4: Hf-Ausgang für Farbfernsehgeräte, kein eingebauter Monitor, komfortable Farbgrafik-Möglichkeiten, eingebaute Tongeneratoren, Anschluß für Kassettenrecorder, Programmiersprache Basic, etwa gleicher Preis.

Während aber der TI-99/4 mit dem 16-bit-Prozessor TMS 9900 arbeitet, baute DAI in sein Gerät die 8-bit-CPU 8085 ein – vor allem, weil DAI seit Jahren mit dieser Prozessorfamilie arbeitet. Nun könnte man meinen, dadurch bestünde ein erheblicher Unterschied in der Verarbeitungsgeschwindigkeit zwichen TI und DAI. Richtig, aber das verhält sich genau umgekehrt, als man erwarten könnte: Der TI-99/4 ist nämlich trotz seiner 16-bit-CPU rund dreimal langsamer bei der Abarbeitung von Basic-Programmen als sein Kollege von DAI. Der Grund: Der TI-99/4-Basic-Interpreter übersetzt nicht direkt in den 9900-Maschinencode, sondern in einen Zwischencode (GPL). Allerdings rechnet der DAI-Computer nur mit sechsstelliger Genauigkeit, was ihn leider für viele finanztechnische Aufgabenstellungen schwierig anwendbar macht.

Da wir gerade bei den Pferdefüßen sind: Die derzeit existierende Dokumentation des DAI-Gerätes kann jemand, der die Unterlagen eines TI-99/4, eines AIM-65, eines PC-100 oder eines TRS-80 kennt, nur als katastrophal bezeichnen. Ohne umfangreiche Vorkenntnisse ist es somit praktisch unmöglich, mit dem Gerät auch nur halbwegs zurechtzukommen – wenn man nicht nur fertige Programme verwenden möchte.

Und bei solchen fertigen Programmen gibt es auch einen Pferdefuß: Zwar ist das Kassetten-Aufzeichnungsformat mit 1200 Bd (Kansas-City) angenehm schnell; es erwies sich aber als deutlich kritischer als z. B. das gleich schnelle Format des AIM-65/PC-100 oder das des PET-2001. Das hat der DAI-Computer also mit dem TRS-80 gemeinsam.



Leider konnte sich DAI auch nicht entschließen, einen Video-Ausgang standardmäßig einzubauen. So muß man in Kauf nehmen, daß die Qualität bei der Wiedergabe über die Antennenbuchse des Farbfernsehgerätes wegen des notwendigen Modulations- und Demodulations-Vorganges besonders bei dem Bildschirmformat von 24 Zeilen mit je 66 Zeichen nicht ganz optimal ist.

Doch nun genug des Negativen! Es gibt am DAI-Computer auch viel Lobenswertes. Ein durchaus brauchbares Monitorprogramm gestattet auch das Progammieren in Maschinensprache – zu solchem Komfort konnten sich in letzter Zeit immer weniger Computerhersteller durchringen; HP-85, TI-99/4 und PC-1000 sind schlechte Beispiele.

Auch braucht man im Gegensatz zum TI-99/4 nicht unbedingt ein Farbfernsehgerät mit NTSC-Norm; der meist schon vorhandene Hausaltar mit deutscher PAL-Norm genügt, um verblüffende Farbgrafiken mit einer Auflösung bis zu 260 x 352 Punkten darzustellen.

Der DAI-Computer ist in RAM-Konfigurationen von 8, 12, 32, 36 oder 48 KByte zu haben. Noch ein kleiner Pferdefuß: Es ist leider nicht möglich, die RAM-Erweiterung selbst vorzunehmen, da man dazu auch ein neues, vom Hersteller programmiertes PROM für die veränderte Adressen-Decodierung benötigt.

An Hardware-Spezialitäten enthält das Gerät eine software-gesteuerte „bank switching"-Logik, die eine Speicherbelegung über die mit 16 Adressenleitungen eigentlich nur erreichbaren 64 KByte hinaus gestattet; eine Logik zum Erkennen eines Stack-Überlaufs, was manches Mal den Computer vor dem „Aufhängen" bei Programmierfehlern bewahrt; einen Hardware-Zufallsgenerator und – auf Wunsch – einen besonderen Arithmetik-Prozessor (AMD 9511), der die CPU entlastet und eine deutliche Geschwindigkeitserhöhung bei mathematischen Aufgaben bringt. Standardmäßig sind drei Tongeneratoren eingebaut, an die ein Stereo-Verstärker angeschlossen werden kann und die dann rechts, links und in der Mitte Toneffekte erzeugen können.

Die Verbindung mit der Außenwelt erfolgt über den leider firmenspezifischen DCE-Bus – hier folgte man dem schlechten Beispiel des TRS-80, der ebenfalls im Grundzustand über keinerlei Eingabe/Ausgabe-Ports verfügt. Dem Benutzer bleibt daher nur übrig, sich die (leider relativ teuren, weil professionellen) DCE-Interface-Karten zuzulegen.

Nicht unerwähnt soll bleiben, daß der DAI-Benutzer den Vorteil einer enorm umfangreichen 8080/8085-Programmbibliothek genießen kann: Assembler, MDS, Fortran sind zwar teuer, aber erhältlich – wie auch die gesamte Intel-8080-Software verwertbar ist.

In unserem kurzen Bericht konnte nur ein Bruchteil der DAI-Besonderheiten erwähnt werden. Deshalb sei noch festgestellt, daß das Gerät den Vergleich mit manch anderem Produkt nicht zu scheuen braucht. Der DAI-Computer ist derzeit wohl einer der preiswertesten mit Farbgrafik-Darstellung und braucht sich auch wegen seiner CPU 8085 nicht unbedingt zu schämen: Dem Basic-Programmierer kann es ziemlich egal sein, ob sein Computer einen 8080, 8085, Z-80 oder 6502 verwendet. Wesentlich ist letztendlich die Kombination zahlreicher herausstechender Eigenschaften zu einem humanen Preis. Fe.

Frank Riedel

Ein bemerkenswerter Taschenrechner

Testbericht über den HP 34 C

Bei interessanten Neuheiten aus dem Bereich Mikrocomputer muß es sich nicht immer um Computer im eigentlichen Sinn handeln – auch bei Taschenrechnern gibt es noch Überraschungen genug. Beispiele dafür sind der HP 41 C von Hewlett-Packard und ein bemerkenswerter „kleiner" Rechner, den HP vor einigen Monaten vorstellte: der HP 34 C.

Es fällt auf, daß die meisten Tasten des HP 34 C vierfach belegt sind. Die Funktionen sind über farbcodierte Präfixtasten zugänglich. Die Tasten haben einen deutlich spürbaren Druckpunkt, und ihre Beschriftung ist trotz der Vierfachbelegung einwandfrei lesbar. Neben den insgesamt nur 34 Tasten sind noch zwei Schiebeschalter für „ON-OFF" und „PRGM-RUN" direkt unterhalb des Displays angeordnet. Das LED-Display ist zehnstellig und gut ablesbar – solange man nicht zu schräg daraufsieht. Angenehm auffallend: Nach jeder dritten Ziffer erscheint ein Dezimalpunkt, der deutlich vom Komma zu unterscheiden ist.

Im Normalbetrieb verfügt der HP 34 C über 21 Speicherregister, ferner über die für die UPN-Logik typischen vier Stack-Register und das „LAST X"-Register. Bei elf Registern ist auch vom Tastenfeld aus Registerarithmetik möglich. Zahlen sind im Bereich von 10^{-99} bis 10^{99} verarbeitbar. Das Anzeigeformat ist schaltbar: Festkomma, wissenschaftlich und technisch mit wählbarer Stellenzahl. Intern rechnet der HP 34 C unabhängig vom Anzeigeformat immer mit voller Genauigkeit. Bei Winkeln stehen drei Anzeigeweisen zur Verfügung. Neben den üblichen mathematischen Funktionen und fünf Statistik-Funktionen hat der Rechner noch ein paar Besonderheiten zu bieten: „INTEGRATE", „SOLVE" und „X!".

„INTEGRATE" berechnet das Integral einer in den Programmspeicher geschriebenen Funktion. Dabei ist die Genauigkeit des Ergebnisses über das Anzeigeformat wählbar, und zusätzlich kann der maximale Fehler angezeigt werden. Man kann also immer einen Kompromiß zwischen Genauigkeit und Rechenzeit schließen. Die Rechenzeit beträgt meist auch bei voller Genauigkeit nur ein paar Sekunden bis zu einigen Minuten. Es gibt jedoch Ausnahmen: Die Berechnung des Integrals

$$\int_0^4 \sqrt{\frac{2}{x} + 2}\, dx$$

ist ein Beispiel dafür. *Tabelle 1* stellt bei verschiedenen Genauigkeiten die Ergebnisse der Rechenzeit gegenüber. Es sei jedoch ausdrücklich betont, daß es sich hierbeit um einen Extremfall handelt. Der Integrand hat in der Umgebung von x=0 eine Polstelle, die mit im Integrationsgebiet liegt. Es ist schon sehr erstaunlich, daß der Rechner in diesem Fall überhaupt ein Ergebnis liefert! Für die Berechnung des gleichen Integrals mit den Grenzen 1 und 4 benötigt der HP 34 C bei voller Genauigkeit lediglich 3 Minuten und 45 Sekunden. Mit „INTEGRATE" lassen sich übrigens auch Integrale berechnen, die nicht geschlossen darstellbar sind, wie z. B. Si(x) (Integralsinus).

„SOLVE" berechnet die reellen Nullstellen einer im Programmspeicher enthaltenen Funktion. Es ist dazu lediglich ein Bereich anzugeben, in dem der Rechner Nullstellen suchen soll. Dies sind aber nur Startwerte; wenn in dem angegebenen Bereich keine Nullstelle liegt, wird auch weitab von diesem Bereich eine Nullstelle gefunden. Bei „SOLVE" wird immer mit voller Genauigkeit gerechnet. Hat ein Polynom mehrere Nullstellen, so kann man die im Programmspeicher enthaltene Funktion immer wieder durch $(x-x_o)$ teilen und dann erneut „SOLVE" verwenden, bis alle reellen Nullstellen gefunden sind. Bei allen Funktionen kann man natürlich auch einfach den Bereich ändern, in dem eine Lösung gesucht werden soll. Auch hier sind analytisch nicht lösbare Aufgaben für den Rechner kein Problem. Beispiel: $e^x - \cos x = 0$. *Tabelle 2* zeigt Ergebnisse und Rechenzeit. Man sieht, daß es wie auch bei „INTEGRATE" ein wenig von den mathematischen Kenntnissen des Benutzers abhängt, welche Ergebnisse man bekommt!

Mit „X!" schließlich lassen sich nicht nur Fakultäten, sondern auch die Gamma-Funktion berechnen. Es ist dabei nur der Zusammenhang x! = I(x+1) zu berücksichtigen. Vor der Berechnung der Gamma-Funktion mit „X!" ist von x also 1 abzuziehen. (Die Gamma-Funktion stellt eine Erweiterung der Fakultät auf nicht ganzzahlige Argumente dar.)

Tabelle 1: Lösungen von $\int_0^{\cdot\cdot} \sqrt{\frac{2}{x} + 2}\, dx$

Anzeigeformat	Ergebnis	Rechenzeit
fix 0	8± 2	15 s
fix 1	8,0 ± 0,2	15 s
fix 2	8,32 ± 0,02	1:55 min
fix 3	8,360 ± 0,002	15:15 min
fix 4	8,3654 ± 0,0002	ca. 4 h

Tabelle 2: Lösungen von $e^x - \cos x = 0$

Berechnete Lösung	Ergebnis mit dieser Lösung	Rechenzeit	Lösungsbereich
0	0	5 s	0
$2{,}593125245 \cdot 10^{-10}$	0	20 s	0,5 bis 1
$-3{,}149877300 \cdot 10^{-12}$	0	2:55 min	50 bis 100
−1,292695719	$-3 \cdot 10^{-10}$	30 s	−1 bis −2

Auch beim Programmieren zeigt der HP 34 C einige erstaunliche Eigenschaften. Er verfügt über eine variable Speicherplatzverteilung, die sich automatisch den Bedürfnissen des Benutzers anpaßt. In der Normalverteilung stehen 70 Programmzeilen und 21 Register zur Verfügung. Bei längeren Programmen werden die Register nacheinander in Programmspeicher umgewandelt, wobei ein Register sieben Programmzeilen entspricht. Auf diese Weise können dann bis zu 210 Programmzeilen benutzt werden, von denen übrigens jede maximal vier Tastendrücke enthalten kann. Die augenblickliche Speicherverteilung kann über die „MEM"-Funktion abgefragt werden. Der HP 34 C benutzt die symbolische Adressierung. Es stehen dafür zwölf Labels zur Verfügung, die auch mehrfach genutzt werden können. Es sind sechs Unterprogrammebenen möglich. Für bedingte Programmverzweigungen kann man neben dem noch zu besprechenden I-Register acht Vergleichsoperationen und vier Flags benutzen. Mit dem äußerst vielseitig verwendbaren I-Register ist die indirekte Adressierung aller Register, Labels und Programmzeilen möglich. Außerdem kann man es mit Hilfe einer „Schleifensteuerungszahl" und den Befehlen „DSE" und „ISG" als Programmzähler verwenden. Ferner ist die indirekte Steuerung des Anzeigeformats möglich. „SOLVE" und „INTEGRATE" lassen sich innerhalb eines Programms auch kombiniert verwenden.

Mit „DEL", „SST" und „BST" kann man Programmzeilen löschen, sich das Programm zeilenweise anschauen oder abarbeiten. Es lassen sich auch ohne weiteres Zeilen einfügen. Bei Bedarf werden dabei sogar automatisch Register in Programmschritte umgewandelt. Probleme mit Sprungadressen treten dabei nicht auf, da ja mit symbolischer Adressierung gearbeitet wird. Neben den Korrekturfunktionen bietet der HP 34 C auch noch eine Art „Bedienerführung": Insgesamt elf Fehlermeldungen sind möglich. Dazu gehört auch eine Selbstprüfroutine. Sämtliche Programme und Registerinhalte bleiben dank Permanentspeicher auch bei ausgeschaltetem Rechner über Monate erhalten. Bei soviel Luxus muß natürlich auch etwas „faul" sein: Der HP 34 C ist relativ langsam, nur etwa halb so schnell wie der bekannte HP 25: Für eine Addition benötigt er rund 0,115 Sekunden.

Abschließend ist noch zu sagen, daß der HP 34 C mit Akkusatz, Netzgerät, Schutzhülle und einer wirklich ausgezeichneten Dokumentation geliefert wird.

µC-Nachrichten

Ab Herbst 1980 ist der neue Mikrocomputer-Lehrgang von Christiani zu haben. Er besteht aus zehn Lektionen nebst zugehörigen Hardware-Lieferungen, kostet etwa 2600 DM und basiert auf dem Mikroprozessor 8085. Eine spätere Erweiterung mit alphanumerischer Tastatur, Video-Interface, Basic-Interpreter, resistentem Assembler, Floppy und A4-Drucker ist vorgesehen.

□

Für Besitzer des 6502-Mikrocomputers SYM-1 gibt es eine eigene Zeitschrift namens „SYM-physis". Zu beziehen ist sie in Europa für 12,50 US-Dollar (Jahres-Abo, 6 Ausgaben) von der SYM Users Group, Postbox 315, Chico, CA 95927, USA.

Streit in den Vereinigten Staaten: An staatlichen Zuschüssen bereichert?

In den Vereinigten Staaten ist unversehens ein Experten-Streit entbrannt, an dem professionelle Mikrocomputer-Anwender und erst recht Hobbyprogrammierer temperamentvoll Anteil nehmen. Es geht um die Frage, ob Software-Unternehmen für Programme, deren Entwicklung durch öffentliche Mittel finanziert wurde, dem Anwender ungerechtfertigt hohe Beträge als Lizenzgebühren und Preis für Dokumente und Datenträger aus der Tasche ziehen.

In der Rolle eines öffentlichen Anklägers tritt in diesem Streit der Fachschriftsteller Adam Osborne auf, und ins Fach des Angeklagten sieht sich nun das Institut für Informationswissenschaften der Universität von Kalifornien in San Diego gedrängt. An diesem Institut entstand die auf Kleinrechnern bisher weitest verbreitete Pascal-Version, UCSD-Pascal (Trademark). Auf das „Trademark", in Deutsch soviel wie „geschütztes Warenzeichen", kommt es in diesem Zusammenhang an: Wenn jemand irgendwo auf der Welt die Sprache UCSD-Pascal verwendet, und sei es auch mit Hilfe eines selbstgestrickten Compilers in einem „Home-Made"-Betriebssystem, dann müßte er von Rechts wegen Lizenzgebühren nach San Diego entrichten.

Keine horrenden Summen zwar (noch unlängst sprach der Direktor Kenneth L. Bowles dieses Instituts in einem Rundfunkinterview von 200 Dollar, die für Software und Dokumentation durch Einzelkunden zu entrichten seien), aber immerhin: Osborne wittert üble Machenschaften, denn er glaubt beweisen zu können, daß die Entwicklung von UCSD-Pascal zumindest zum Teil durch immense Staatszuschüsse ermöglicht wurde. In der Februar-Ausgabe von „Interface Age" äußert Osborne nun die Vermutung, es könnten auch noch andere Hochschulen oder Firmen auf anfechtbare Weise ihre Entwicklungsergebnisse versilbern – und denen gibt Osborne einen grimmigen Rat: Sie möchten doch schleunigst offenlegen, in welchen ihrer Verkaufsprogramme Steuergelder stecken, ehe er, Adam Osborne, diesen Anbietern auf die Schliche kommt.

Wie kommt es nun, daß der Schriftsteller mit einer so reputierlichen Einrichtung wie dem berühmten Institut zu San Diego Händel sucht? Eingeweihte glauben es zu wissen. Auch Osborne entwikkelt und verkauft Programme, freilich: einen Staatszuschuß soll die Firma „Osborne and Associates" noch nie erhalten haben – und so etwas schärft natürlich den Blick für die Ungerechtigkeiten dieser Welt.

Der europäische Beobachter blickt halb belustigt, halb in Traurigkeit über den großen Teich: Staatszuschüsse zur Förderung der Entwicklung von Mikrocomputer-Programmen, die dann – wie UCSD-Pascal – jedermann gegen mäßige Gebühr zugänglich sind? Ach, wie wäre das schön, wenn es so etwas auch bei uns gäbe. Aber hierzulande wurden in der Vergangenheit die Software-Millionen des Staates an nationale Einrichtungen wie Siemens vergeben, damit man dort nachvollziehen konnte, was Firmen wie IBM schon längst entwickelt hatten. Amerika, du hast es besser...

Rudolf Hofer

BASIC vor dem Handbuch

2. Teil

Im ersten Teil wurde bereits erläutert, wie Programme auf Kassetten „konserviert" werden. Diesmal geht es um Datensicherheit beim Abspeichern. Außerdem wird anhand eines Sortierprogramms der Umgang mit String-Variablen gezeigt.

Eine bewährte Methode, die „Datensicherheit" zu erhöhen, ist es, jedes Programm zweimal hintereinander aufzunehmen. Schlechte Bandstellen können nämlich dazu führen, daß Programme unvollständig auf das Band gebracht werden. Will man jedoch sicher gehen, lädt man jedes abgespeicherte Programm nochmal mit CLOAD? Es wird dann geprüft, ob der Speicherinhalt mit den geladenen Daten identisch ist. Stimmen sie nicht überein, kommt eine Fehlermeldung. Ist alles in Ordnung, meldet sich der Computer mit READY.

Das Laden eines Programmes geht so vor sich:
Bandstelle mit dem gewünschten Programm suchen.
CLOAD oder CLOAD „A" eingeben (anstelle von A steht der Kennbuchstabe des gewünschten Programms). Ohne den Kennbuchstaben wird einfach das nächste Programm geladen. Ist er vorhanden, werden alle Programme mit einer anderen Kennung übergangen.
Kassettenrecorder auf Wiedergabe.

Jetzt wird das Programm wieder in den Computer geladen. Steht noch eins im Speicher, so löscht es die Maschine, auch wenn sich die Zeilennummern des alten und des neuen Programms nicht überschneiden. Der Eingabevorgang ist für CLOAD? identisch. Allerdings wird bei diesem Kommando das im Speicher stehende Programm nicht gelöscht, es soll ja mit dem Kassetteninhalt verglichen werden.

Es empfiehlt sich, eine genaue Buchführung über die abgespeicherten Programme und die Bandstellen, an denen sie sich befinden, zu führen. Am besten fügt man auch eine Kurzbeschreibung des Programms bei. Nur dann findet man auf Anhieb, was man sucht.

An das Bandmaterial werden keine besonderen Ansprüche gestellt. Man sollte allerdings keine zu langen Bänder verwenden, da sonst die Suche bestimmter Abschnitte sehr viel Zeit verschlingt. Ideal sind spezielle Hobbycomputer-Bänder, die auf jeder Seite ungefähr sieben Minuten laufen, das entspricht etwa einer Kapazität von 25 000 Zeichen.

Fehlermeldungen

Computer sind keine intelligenten Wesen. Im Grunde sind sie sogar sehr dumm. Wenn man ihnen z.B. sagt GO TO ZEILE 100, dann begreifen sie gar nichts, obwohl es doch eindeutig ist, was man will. Die Maschine versteht uns aber nur dann, wenn wir uns bis auf das letzte i-Tüpfelchen an die vorgegebenen Regeln halten. Und die besagen nun mal, daß nach GOTO nur eine Zeilennummer steht.

Diese formalen Regeln nennt man die Syntax der Sprache (übrigens nicht nur bei Computersprachen). Gibt man dem Computer Kommandos oder Befehle ein, die der Syntax widersprechen, dann gibt er schlicht ERROR oder SYNTAX ERROR aus. Oft gibt er dem Benutzer auch noch nähere Hinweise, was er falsch gemacht hat und wo. In diesem Fall bleibt uns im Augenblick nur, das Programm genauestens zu überprüfen und an den verdächtigen Stellen zu ändern. Die Hinweise auf Syntaxfehler werden in den Handbüchern erklärt. Hier unterscheiden sich die verschiedenen Modelle erheblich.

Syntaxfehler darf man nicht mit logischen Fehlern verwechseln: Wenn das Programm keine Syntaxfehler mehr hat, kann es trotzdem falsche Ergebnisse liefern. Das liegt aber dann an einer falschen oder unvollständigen Grundüberlegung.

Der Computer verarbeitet nicht nur Zahlen

Wir kommen jetzt zu einem ganz wesentlichen Punkt, der den Computer von Tisch- und Taschenrechnern unterscheidet: Er kann neben Zahlen auch Text verarbeiten. Daß er Texte ausdrucken kann, haben wir bereits gesehen.

In der mit englischen Ausdrücken überschwemmten Computer-Fachsprache nennt man Texte im allgemeinen Strings. Obwohl es für gewöhnlich nicht einzusehen ist, daß man ein gleichwertiges deutsches Wort durch ein englisches ersetzt, wollen wir in diesem Fall „String" benutzen, da es dann nicht zu Verwechslungen z.B. mit BASIC-Text (= Programm) kommen kann. Ein String ist im weiteren eine Folge von max. 255 Buchstaben, Zahlen oder Sonderzeichen einschließlich des Leerraums. Verboten sind Kommas innerhalb eines Strings.

Wir haben bisher nur numerische Variablen kennengelernt. Ein neuer Typ sind die sogenannten String-Variablen, denen als Wert ein String zugewiesen wird. Sie werden wie numerische Variablen bezeichnet, erweitert um ein Dollarzeichen, z.B. A$, B$, X$ usw. Das Beispiel von *Bild 5* macht deutlich, wie sie benutzt werden.

In Zeile 110 steht beim INPUT-Befehl jetzt eine String-Variable. Das Programm druckt ein Fragezeichen aus und hält so lange an, bis der Benutzer einen Text eingetippt hat. Natürlich kann dieser Text rein theoretisch auch aus Zahlen bestehen, allerdings könnte der Computer damit nicht rechnen – das kann er nur mit numerischen Variablen. Da im Beispiel nach einem Namen gefragt wird, nehmen wir an, wir hätten mit KARL geantwortet. Von jetzt ab hat die Variable

N$ den Wert KARL. Ähnlich wie bei numerischen Variablen ist auch folgende Zuweisung möglich:
A$ = KARL
Zu beachten ist lediglich, daß der String in Anführungszeichen stehen muß (innerhalb des Strings kann in diesem Fall natürlich kein Anführungszeichen erscheinen).

Doch weiter im Beispiel: Nachdem der Computer nach der Anzahl der gerauchten Zigaretten gefragt hat, erwartet er die Eingabe einer Zahl, die er der numerischen Variablen Z zuweist.

Zeile 140 besagt: Wenn Z größer als 10 ist, dann gehe zu Zeile 200, anderenfalls mache bei 150 weiter. In den Zeilen 150 und 200 stehen PRINT-Befehle, wie sie uns schon bekannt sind. Neu ist lediglich, daß auch der Wert von String-Variablen ausgedruckt wird. Anstelle von N$ wird deshalb KARL ausgedruckt, bzw. der Text, den wir beim INPUT-Befehl eingetippt haben. Auf diese Weise fügt der Computer z.B. Namen und Adressen in Formbriefe ein. END in Zeile 160 beendet das Programm, da sonst Zeile 200 noch ausgeführt wird.

Die wichtigste Operation, die mit String-Variablen ausgeführt werden kann, ist der Vergleich.
Beispiel:
100 IF A$ = B$ THEN 200

Es kann aber auch gefragt werden, ob ein String größer, größer-gleich, kleiner oder kleiner-gleich ist. Die Vergleichsoperatoren entsprechen denen für numerische Variable (>, >=, <, <=). Aber was bedeutet, ein String ist größer als ein anderer?

Dazu muß man folgendes wissen: Im Computer werden Buchstaben, Ziffern und Sonderzeichen wie Zahlen dargestellt. *Bild 6* zeigt ihre Reihenfolge. Das bedeutet, daß A kleiner als B ist, B kleiner als C usw. Sinnvollerweise ist jeder Buchstabe, der im Alphabet weiter vorne steht kleiner, als einer, der hinter ihm rangiert. Auch die Ziffern und Sonderzeichen haben eine festgelegte Reihenfolge.

Solange wir Strings vergleichen, die aus einem Buchstaben bestehen, ist alles klar. Was aber passiert, wenn ganze Wörter verglichen werden? Kommt es auf die Zahl der Zeichen an, oder werden alle Zeichen bewertet und addiert? Nein! Es ist viel einfacher und nützlicher:

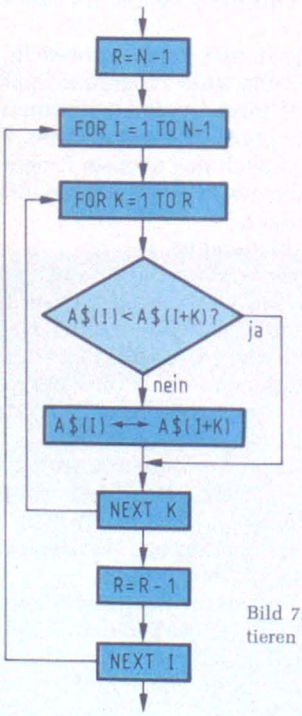
Der Computer vergleicht zunächst die ersten Buchstaben. Sind sie unterschiedlich, dann ist für ihn der String kleiner, dessen erster Buchstabe kleiner ist. Sind beide gleich, dann geht er zum nächsten und vergleicht ihn mit dem zweiten Buchstaben des anderen Strings. Diese Prozedur wiederholt sich, bis sich die Buchstaben unterscheiden. Er geht also genauso vor, wie man einen Namen im Telefonbuch sucht.

```
100 PRINT "WIE HEISSEN SIE";
110 INPUT N$
120 PRINT "WIEVIELE ZIGARETTEN RAUCHEN SIE PRO TAG, ";N$;"?"
130 INPUT Z
140 IF Z>10 THEN 200
150 PRINT "BRAVO ";N$;",";Z;" ZIGARETTEN SIND NICHT ZU VIEL"
160 END
200 PRINT N$;", SIE LEBEN SEHR UNGESUND"

WIE HEISSEN SIE? CHARLIE
WIEVIELE ZIGARETTEN RAUCHEN SIE PRO TAG, CHARLIE?
5
BRAVO CHARLIE, 5  ZIGARETTEN SIND NICHT ZU VIEL

WIE HEISSEN SIE? KARL
WIEVIELE ZIGARETTEN RAUCHEN SIE PRO TAG, KARL?
20
KARL, SIE LEBEN SEHR UNGESUND
```

Bild 5. Programm mit Ausführungsbeispielen zur Demonstration von String-Variablen ($ ≙ $, auch in allen folgenden Ausdrucken)



Vorher: A$(1) = BERTA
A$(2) = DORA
A$(3) = ANTON
A$(4) = CAESAR

Nach dem 1. Schritt:
A$(1) = ANTON
A$(2) = DORA
A$(3) = BERTA
A$(4) = CAESAR

Nach dem 2. Schritt:
A$(1) = ANTON
A$(2) = BERTA
A$(3) = DORA
A$(4) = CAESAR

Nachher: A$(1) = ANTON
A$(2) = BERTA
A$(3) = CAESAR
A$(4) = DORA

Bild 7. Flußdiagramm: Sortieren von Namen

Hex-Code der ASCII-Zeichen

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENG	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	CS	RS	VS
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	–	.	/
3	Ø	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	–
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	'	}	~	DEL

◀ Bild 6. Darstellung von Zeichen im Computer, z. B. „A" = hex 41 bzw. dezimal 65

In einer alphabetisch geordneten Namensliste ist demnach jeder Name (String), der weiter vorne erscheint, kleiner als einer hinter ihm.

Diese Tatsache kann man natürlich auch umgekehrt verwenden. Wenn man dem Computer den Auftrag gibt „Drucke eine Folge von Strings in der Reihenfolge ihrer Größe", dann ordnet er Begriffe nach dem Alphabet. Ein solches Programm wollen wir jetzt schreiben.

Wir sortieren Namen

Damit wir dieses Problem lösen können, müssen wir uns vorher mit den „indizierten Variablen" auseinandersetzen. Sie werden genauso benutzt wie die nichtindizierten Variablen, mit denen wir bisher gearbeitet haben. Der Unterschied besteht darin, daß sie mit einem Zusatzmerkmal, dem Index, versehen sind.

Beispiele: A(1), B$(3), X$(I), Y(J+K)

Wir erkennen, daß der Index einfach in Klammern angefügt wird. Er kann aus einer simplen Zahl oder aus einem Ausdruck bestehen. Natürlich ist es auch möglich, daß eine einzige Variable als Index fungiert. Dieser Fall tritt sehr häufig auf.

Der Sinn von indizierten Variablen wird unmittelbar klar, wenn man folgendes Beispiel betrachtet: Eine bestimmte Anzahl von Namen soll in den Computer eingegeben werden. Bei Bedarf soll die Möglichkeit bestehen, sie wieder auszudrucken.

Mit unseren bisherigen Kenntnissen würden wir etwa folgendermaßen vorgehen:

Eingabe

```
10 INPUT A,B,C,D,E
```

Ausgabe

```
100 PRINT A,B,C,D,E
```

Das sieht ganz einfach aus. Aber was passiert, wenn wir 20 oder noch mehr Namen einspeichern wollen? Wir müßten so viele verschiedene Variablen-Namen benutzen, wie Namen eingegeben werden: Ein recht unhandliches Verfahren.

Mit Hilfe einer indizierten Variablen geht die Eingabe so:

```
100 FOR I = 1 TO 5
110 INPUT A$(I)
120 NEXT I
```

Starten wir das Programm jetzt mit RUN, werden I der Reihe nach die Werte 1...5 zugewiesen. Demzufolge erfragt der Computer in Zeile 110 vom Benutzer der Reihe nach den Wert der String-Variablen A$(1)...A$(5), also die gewünschten Namen. Sollen diese Namen im Verlauf des Programms wieder ausgedruckt werden, genügt dazu die

Befehlsfolge

```
200 FOR I = 1 TO 5
210 PRINT A$(I)
230 NEXT I
```

Natürlich könnten wir auch die Befehle

```
200 FOR K = 1 TO 5
220 PRINT A$(K)
230 NEXT K
```

verwenden. Wichtig ist allein, daß der Index die Werte 1...5 durchläuft.

Wie leicht einzusehen ist, kann das Programm jetzt auf einfache Weise für mehr Namen abgeändert werden. Dazu ist lediglich die Zahl 5 in Zeile 100 entsprechend zu ändern. Bei der Ausgabe ist dann natürlich auch die 5 in Zeile 200 abzuändern.

Noch eleganter kann man das Problem lösen, wenn man die Anzahl der Namen variabel macht.

```
100 PRINT "WIEVIELE NAMEN"
110 INPUT N
115 REM EINGABE
120 FOR I = 1 TO N
130 INPUT A$(I)
140 NEXT I
150 REM AUSGABE
200 FOR I = 1 TO N
210 PRINT A$(I)
220 NEXT I
```

Zugleich haben wir den einfachsten aller BASIC-Befehle eingeführt: REM. Der REM-Befehl dient lediglich dazu, das Programm mit Kommentaren zu versehen. Er bewirkt keine Aktion.

Wir wollen nun aber auf unser eigentliches Problem zurückkommen: das Sortieren von Namen. Wie die Namen eingegeben werden, haben wir soeben gelernt. Auch die Ausgabe bereitet uns keine Schwierigkeiten mehr. Es bleibt also lediglich die Aufgabe, den Variablen, in diesem Fall A(1)...A(N), die Namen so zuzuweisen, daß A(1) den „kleinsten" String erhält, A(2) den „zweitkleinsten" usw.

Keine Angst, wenn Sie die Lösung, die wir im folgenden in alle Bestandteile zerlegen wollen, nicht auf Anhieb voll durchschauen. Das Problem ist nämlich durchaus nicht so trivial, wie es zunächt den Anschein hat. Sollten Sie aber in dieser Aufgabe keine ernsthaften Schwierigkeiten sehen, haben Sie bereits eine schwere Hürde mit Bravour genommen.

Bild 7 zeigt den Ablauf des Sortiervorgangs, unter der Voraussetzung, daß eine Namensliste in beliebiger Reihenfolge den Variablen A$(1)...A$(N) zugeordnet wurde. N hat also vor dem Eintritt in das Sortierprogramm den Wert 4. Um den Vorgang zu veranschaulichen, wollen wir einen kompletten Programmdurchlauf anhand der in *Bild 7* abgedruckten Namensliste durchspielen:

```
1. Schritt:  I=1, K=1, N=4, R=3
             A$(1)<A$(2)?, ja weil BERTA<DORA
             K=R?, nein
             K=K+1=2
             A$(1)<A$(3)?, nein weil BERTA>ANTON
             A$(1)=ANTON, A$(3) BERTA
             K=R?, nein
             K=K+1=3
             A$(1)<A$(4)?, ja weil ANTON<CAESAR
             K=R?, ja
             R=R–1=2
             I=N–1?, nein
             Sprung zum Programmanfang
```

Bei diesem Schritt passierte folgendes: A$(1) wurde der Reihe nach mit den folgenden Variablen A$(2), A$(3) usw. verglichen. Wurde eine Variable gefunden, die kleiner war (also an früherer Stelle im Alphabet steht), dann wurde ihr Wert mit dem von A$(1) vertauscht. Als Ergebnis steht der kleinste Wert der gesamten Liste nun an erster Stelle. Der Rest der Liste ist noch ungeordnet, muß also noch in die richtige Reihenfolge gebracht werden. A$(1) kann aber künftig außer Acht gelassen werden. Dasselbe Verfahren wird nun auf die Variablen A$(2)...A$(N) angewendet. Dazu wird der Index I zunächst am Programmanfang um eins erhöht. Die Bedeutung von K ist bereits erkennbar. Während I bei jedem Schritt fest bleibt, wird K so verändert, daß man der Reihe nach mit den Variablen vergleichen kann, deren Index höher ist. Nun wird die Restliste, die noch zu bearbeiten ist aber immer kürzer, deshalb wird auch der höchste Wert von K, der mit R bezeichnet ist, immer kleiner. *(Schluß folgt)*

Lineare Gleichungssysteme

Das Programm im *Bild* berechnet die Lösung linearer Gleichungssysteme mit bis zu 15 Gleichungen mit 15 Unbekannten und führt die Inversion von Matrizen bis zur maximalen Größe von 15x15 durch. Die Beschränkung auf 15 Gleichungen begründet sich *nicht* durch den zur Verfügung stehenden Speicherraum des PET, sondern durch einen Softwarefehler. Es können zwar Felder mit mehr als 256 Elementen dimensioniert werden, jedoch wird z. B. bei Änderung des 256-ten Elementes das 0-te Element ebenfalls geändert. Bei Systemen, die diesen Fehler nicht aufweisen, sind durchaus Gleichungssysteme mit mehr als 30 Unbekannten zu lösen (in 8 K). Der zur Lösung verwendete Gauß-Jordan-Algorithmus transformiert das Gleichungssystem:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= a_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= a_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= a_n \end{aligned} \qquad (1)$$

in ein System der Form:

$$\begin{aligned} b_{11}x_1 &= b_1 \\ b_{22}x_2 &= b_2 \\ &\vdots \\ b_{nn}x_n &= b_n \end{aligned} \qquad (2)$$

Hierzu wird die erste Gleichung mit

$$-\frac{a_{i1}}{a_{11}}$$

multipliziert (i = 2,...,n; mit n = Anzahl der Unbekannten = Anzahl der Gleichungen) und jeweils zur i-ten Gleichung addiert (Gauß-Schritt). Man erhält dann ein Gleichungssystem der Form:

$$\begin{aligned} c_{11}x_1 + c_{12}x_2 + \ldots + c_{1n}x_n &= c_1 \\ c_{22}x_2 + \ldots + c_{2n}x_n &= c_2 \\ &\vdots \\ c_{n2}x_2 + \ldots + c_{nn}x_n &= c_n \end{aligned} \qquad (3)$$

wobei

$c_{1k} = a_{1k}$, k = 1...n

$c_1 = a_1$

Jetzt wird die zweite Gleichung mit

$$-\frac{c_{i2}}{c_{22}}$$

(i = 1, 3, 4, ..., n) multipliziert und zu den restlichen Gleichungen addiert (1. Jordan-Schritt). Man erhält:

```
100 PRINT"sR***LINEARE GLEICHUNGSSYSTEME***r
110 POKE59468,14:REM DIETER SMODE, 10-12-1978
120 PRINT"QDieses Programm loest lineare Glei-":PRINT"chungssysteme.
130 PRINT"QDie Berechnung der inversen Matrix kann"
140 PRINT"durch Eingabe von N=0 und anschliessen-":PRINT"de";
150 PRINT" Eingabe des richtigen N erfolgen.":PRINT"Die Berechnung erfolgt ";
160 PRINT"spaltenweise!":PRINT"QN darf maximal gleich 15 sein!!":PRINT
170 CLR:D=1:INPUT"N=";N:IFN=0THEN360
180 N=N-1:IFN>14THENPRINT"q            ":PRINT"qq":GOTO170
190 DIMA(N,N+1):PRINT"QKoeffizientenmatrix"
200 FORI=0TON:FORJ=0TON+1:IFJ=N+1THENPRINT"QA(";I+1;")=";:INPUTA(I,J):GOTO220
210 PRINT"A(";I+1;",";J+1;")=";:INPUTA(I,J)
220 NEXTJ:PRINT:PRINT:NEXTI
230 PRINT:PRINT:FORK=0TON:IFK=NTHEN270
240 AB=ABS(A(K,K)):FORI=K+1TON:IFABS(A(I,K))<=ABTHEN260
250 AB=ABS(A(I,K)):D=-D:FORJ=KTON+1:Z=A(K,J):A(K,J)=A(I,J):A(I,J)=Z:NEXTJ
260 NEXTI
270 IFABS(A(K,K))<1E-7THEN340
280 FORI=0TON:IFI=KTHEN300
290 F=A(I,K)/A(K,K):FORJ=K+1TON+1:A(I,J)=A(I,J)-A(K,J)*F:NEXTJ
300 NEXTI,K:IFC<>0THEN430
310 FORI=0TON:D=D*A(I,I):NEXTI:PRINT"DET(A)=";D:PRINT"QLoesungsvektor"
320 FORI=0TON:PRINT"X";I+1;"=";A(I,N+1)/A(I,I):NEXTI:PRINT
330 PRINT"**********************************Q":GOTO170
340 PRINT"Gleichungssystem nicht eindeutig loesbar":PRINT"A(";K+1;",";K+1;")";
350 PRINT" verschwindet annaehernd!":GOTO330
360 PRINT"QQBerechnung der inversen MatrixQ":INPUT"N=";N
370 N=N-1:IFN>14THENPRINT"q            ":PRINT"qqqqqq":GOTO360
380 DIMA(N,N+1):DIMD(N,N+1):PRINT"QKoeffizientenmatrixQ"
390 FORI=0TON:FORJ=0TON:PRINT"D(";I+1;",";J+1;")=";:INPUTD(I,J)
400 NEXTJ:PRINT:NEXTI
410 FORI=0TON:FORJ=0TON:A(I,J)=D(I,J):NEXTJ,I
420 J=N+1:FORI=0TON:A(I,J)=0:NEXTI:A(C,J)=1:C=C+1:GOTO230
430 PRINTC;". Spalte":PRINT:FORI=0TON:PRINT"A(";I;",";C;")=";A(I,N+1)/A(I,I)
440 NEXTI:PRINT
450 IFC<=NTHENPRINT"WEITER?":GOTO470
460 GOTO170
470 GETW$:IFW$=""THEN470
480 IFW$="J"THEN410
490 GOTO170
```

Basic-Programm in der PET-Syntax zum Lösen von bis zu 15 Gleichungen mit 15 Unbekannten

$$
\begin{aligned}
d_{11}x_1 &+ d_{13}x_3 + \ldots + d_{1n}x_n = d_1 \\
d_{22}x_2 &+ d_{23}x_3 + \ldots + d_{2n}x_n = d_2 \\
&+ d_{33}x_3 + \ldots + d_{3n}x_n = d_3 \\
&\quad\vdots \qquad\qquad\quad \vdots \\
&+ d_{n3}x_3 + \ldots + d_{nn}x_n = d_n
\end{aligned}
\tag{4}
$$

wobei
$k = 2 \ldots n$
$d_{2k} = c_{2k}$,
$d_2 = c_2$

Der obige Schritt von (3) nach (4) wird jetzt mit der dritten Gleichung wiederholt (mit $-d_{i3}/d_{33}$ multiplizieren, i = 1, 2, 4, ... n), so daß nur der in der dritten Gleichung vor x_3 stehende Koeffizient von Null verschieden ist (2. Jordan-Schritt).

Nach n-1 Jordan-Schritten erhält man schließlich das Gleichungssystem (2), dessen Lösungen durch einfache Divisionen berechnet werden können.

Um Rundungsfehler möglichst klein zu halten, werden vor einem Gauß- bzw. Jordan-Schritt die Gleichungen so vertauscht, daß jeweils die Gleichung mit dem betragsgrößten Koeffizienten von x_k zur Elimination von x_k in den restlichen Gleichungen benutzt wird. Dies würde z. B. beim Gauß-Schritt bedeuten, daß die Gleichungen so vertauscht werden müssen, daß gilt:

$$|a_{11}| \geqq |a_{k1}|,\ k = 2, 3, \ldots n \tag{5}$$

Die Elimination und auch die Berechnung der Determinanten wird von den Zeilen 230...320 durchgeführt, wobei die Determinante wie folgt berechnet wird:

$$\det A = (-1)^m \cdot b_{11} \ldots b_{nn} \tag{6}$$

m = Anzahl der Vertauschungen

Bei jeder Vertauschung wird $(-1)^m$ dies durch D = –D berücksichtigt.

Eine weitere Anwendung für dieses Programm ist die Matrizeninversion. Wenn in (1) gilt:

$$a_i = 0, \text{ für } i = 1, 2, \ldots k-1, k+1, \ldots n \tag{7}$$
$$a_k = 1$$

ergibt sich als Lösung des Gleichungssystems die k-te Spalte der inversen Koeffizientenmatrix.

Dieses Verfahren wird hier zur Berechnung der Inversen einer Matrix angewendet (Zeilen 390...490), so daß die Berechnung spaltenweise erfolgt.

Von geringfügigen Änderungen abgesehen, dürfte dieses Programm auf jedem Basic-Rechner laufen. Die Rechenzeit für ein Gleichungssystem mit 15 Unbekannten liegt unter einer Minute.

Literatur

[1] *Jordan-Engeln, G.; Reutter, F.:* Numerische Mathematik für Ingenieure. BI-Hochschultaschenbuch 104

[2] *Jordan-Engeln, G.; Reutter, F.:* Formelsammlung zur numerischen Mathematik mit Fortran-IV-Programmen. BI-Hochschultaschenbuch 106

Mehr Standard-Texte bei RTTY

Das in [1] und [2] veröffentlichte Programm wurde mit Erfolg eingesetzt; etwas störend war, daß es nur den Abruf eines einzigen Standardtextes gestattet. Die Programmerweiterung im *Bild* erlaubt den Abruf von max. sieben Texten, z. B. RYRYRY-Prüftext, CQ-Ruf, Quick Brown Fox usw.

Zum Abruf der einzelnen Textblöcke dienen die im Urprogramm ungenutzten Tasten F1, ", #, $, %, *, >. Voraussetzung dazu ist, daß der RAM-Umfang mindestens 2 K umfaßt. Die mit * gekennzeichneten Speicherinhalte sind abhängig von der Anfangslage der einzelnen Textstücke und hier als Beispiel für einen bestimmten Anwenderfall angegeben. Sie lassen sich jedoch sehr leicht an individuelle Bedürfnisse anpassen, wenn man, wie in Heft [3] beschrieben, mit dem Editor-Befehl nacheinander ab 0500 die Texte eingibt und mit dem M-Befehl unter E1 jeweils das Ende aufsucht, das mit 0D und 00 abgeschlossen ist. Die Adresse des darauffolgenden Speicherplatzes enthält in ihrem niederwertigen Byte den Inhalt der anzupassenden Speicherzellen (042F, 0434, 0439, 043E) und entspricht der Startadresse des nachfolgenden Textes innerhalb der 5. Page.

Bei der Texteingabe braucht hier nicht die Endadresse eingegeben zu werden – nach dem Erscheinen von „TO" einfach zweimal „Space" tasten. Es ist zu beachten, daß die ersten fünf Texte innerhalb der 5. Page untergebracht sein müssen. Die Textblöcke 6 und 7 können jeweils eine ganze Page einnehmen und erlauben daher eine relativ große Länge. Hier müssen die Anfangsadressen nicht angepaßt werden, da sie durch das Programm bereits auf 0600 bzw. 0700 festgelegt sind.

Eine Kombination mit dem Sprachsynthese-Rufzeichengeber [4] ist leider nicht ohne weiteres möglich, da die Standardtexte von den digitalisierten Sprachsignalen überschrieben würden.

Werner Lang, DG 4 AQ

Literatur

[1] AIM empfängt RTTY. FUNKSCHAU 1979, Heft 15, Seite 884.

[2] AIM sendet RTTY. FUNKSCHAU 1979, Heft 16, Seite 953.

[3] Nachtrag zu „AIM sendet RTTY". FUNKSCHAU 1980, Heft 2, Seite 82.

[4] RTTY mit Sprachsynthese. FUNKSCHAU 1980, Heft 16, Seite 85.


```
023B 4C JMP 0400
023E EA NOP
  :   :   :
0242 EA NOP

025F BD LDA 0500,X

0400 A2 LDX #05
0402 8E STX 0261
0405 C9 CMP #0D
0407 F0 BEQ 0428
0409 C9 CMP #5B
040B F0 BEQ 042B
040D C9 CMP #22
040F F0 BEQ 042E
0411 C9 CMP #23
0413 F0 BEQ 0433
0415 C9 CMP #24
0417 F0 BEQ 0438
0419 C9 CMP #25
041B F0 BEQ 043D
041D C9 CMP #2A

041F F0 BEQ 0442
0421 C9 CMP #3E
0423 F0 BEQ 044A
0425 4C JMP 0243
0428 4C JMP 0230
042B 4C JMP 0258
042E A2 LDX #10 *
0430 4C JMP 025A
0433 A2 LDX #44 *
0435 4C JMP 025A
0438 A2 LDX #82 *
043A 4C JMP 025A
043D A2 LDX #EA *
043F 4C JMP 025A
0442 A2 LDX #06
0444 8E STX 0261
0447 4C JMP 0258
044A A2 LDX #07
044C 8E STX 0261
044F 4C JMP 0258
```

Bis zu sieben Standardtexte kann man beim AIM-65-Funkfernschreib-Programm nun abrufen, wenn man diese kleine Programmergänzung eintippt

π auf 200 Stellen

Bereits häufiger war von der Rechengenauigkeit des PET in der FUNKSCHAU die Rede. Die Ursachen des widersprüchlichen Verhaltens bei Kurzschreibweise und ausgeschriebener Form sind inzwischen hinreichend geklärt (siehe Sonderheft Hobbycomputer 2). Trotzdem ergeben sich an manchen Stellen für den PET-Benutzer Probleme bezüglich der Rechengenauigkeit des PET. Am Beispiel der Bestimmung von π (3,14...) soll nun ein Weg gezeigt werden, wie sich die Rechengenauigkeit des PET auf nahezu beliebig viele Stellen steigern läßt.

Zur Bestimmung des Wertes von π gibt es relativ viele brauchbare Verfahren. Am bekanntesten dürfte die Monte-Carlo-Methode sein; hierbei wird untersucht, ob ein zufällig erzeugter Punkt innerhalb, außerhalb oder auf einem Punkt des Einheitskreises in einem Viertelquadranten liegt.

Ein weiteres Verfahren besteht darin, daß ein Kreis von innen und von außen solange in quadratische Flächen zerlegt wird, bis beide Rechenverfahren den gleichen Wert ergeben (Fläche innen = π = Fläche außen).

Im Jahre 1940 gaben Simon und Schuster einen Algorithmus zur Lösung des Problems an, der bei hinreichender Annäherung an unendlich gegen π konvergiert. Die verwendete Form des Algorithmus lautet:

$$\pi = \sum_{k=1}^{\infty} \frac{16\,(-1)^{k+1}}{(2\,k-1)\,5^{2k-1}} - \sum_{k=1}^{\infty} \frac{4\,(-1)^{k+1}}{(2\,k-1)\,239^{2k-1}}$$

Bei der Beschränkung des PET auf acht Stellen sowie auf die Rechengenauigkeit im Bereich von −1E−37 und 1E37 läßt sich dieser Algorithmus nicht mehr direkt eingeben. Es mußte also eine Möglichkeit gefunden werden, die Einzelglieder des Algorithmus PET-gerecht zu gestalten.

Behindert durch das alte Betriebssystem (Felder nur bis 256 Elemente dimensionierbar) sowie durch den geringen Speicherplatz werden alle Variablen direkt in die Speicherregister „gepoked" und über PEEK wieder ausgelesen *(Bild)*. In der 8-K-Version des PET, für den dies Programm geschrieben wurde, ergibt sich durch diesen Trick die Möglichkeit, π auf 200 Nachkommastellen zu berechnen. Bei der Verwendung eines PET mit mehr Speicherplatz muß folgende Änderung vorgenommen werden, um mehr Stellen berechnen zu können:

In Zeile 50 sind die Anfangsadressen für die Variablenfelder definiert. Bei z. B. 24-K-RAM-Bereich könnte Zeile 50 lauten:
50 c(1) = 25 : C (2) = 239 : PW = 2900 : TE = 8000 : R = 10900
Einzig zu beachten ist, daß immer PW + TE = R gilt. Da das Programm in Basic geschrieben wurde, sollte es ohne Veränderungen (mit Ausnahme der Feldadressen im RAM) auch auf anderen Maschinen laufen. Ein Nachteil soll hier allerdings nicht verschwiegen werden: Zur Berechnung von 1000 Nachkommastellen benötigt der PET runde 26 (sechsundzwanzig!) Stunden, der Apple braucht ziemlich genau 2 Tage.

Michael Penzkofer

Two π or not two π – that's the question...

SHAKESPEARE

```
 15 PRINT"sQQQWIEVIELE ZIFFERN VON π SOLLEN BERECHNET WERDEN ?
 20 INPUTSI:T=10:IFSI>200THEN50
 30 T=100:SI=(SI+1)/2
 50 C(1)=25:C(2)=239:PW=2900:TE=5000:R=7900
 125 FORPA=1TO2:GOSUB4000
 200 GOSUB5000
 210 PT=TE:DI=E
 215 GOSUB1000
 220 IFSN>0THENGOSUB2000
 230 IFSN<0THENGOSUB3000
 240 E=E+2:SN=-SN
 250 PT=PW:DI=C(PA):GOSUB1000
 260 IFPA=2THENGOSUB1000
 270 IFZE<>0THEN200
 300 NEXTPA
 500 PRINT"sQQQDER WERT VON π AUF "(T/100+1)*SI-1;" DEZIMALSTELLEN : ":PRINT
 510 FR=PEEK(R)
 520 PRINTFR;"|. ";
 530 FORPL=R+1TOR+SI
 540 IFT=10THEN570
 560 IFPEEK(PL)<10THENPRINT"0";
 570 PRINT"|"PEEK(PL);
 580 NEXTPL
 590 PRINT:END
 1000 REM
 1020 D=0:ZE=0:FORPL=PTTOPT+SI
 1030 D=D+PEEK(PL)
 1040 QU=D/DI
 1050 RS=D-INT(DI)*INT(QU)
 1055 ZE=ZEOR(RS+QU)
 1060 POKEPL,QU
 1070 D=T*RS:NEXTPL
 1080 RETURN
 2000 CA=0:FORPL=SITO0STEP-1
 2030 SU=PEEK(R+PL)+PEEK(TE+PL)+CA
 2050 CA=0:IFSU<TTHEN2080
 2060 SU=SU-T:CA=1
 2080 POKER+PL,SU
 2090 NEXTPL:RETURN
 3000 LO=0:FORPL=SITO0STEP-1
 3030 DC=PEEK(R+PL)-PEEK(TE+PL)-LO
 3050 LO=0:IFDC>=0THEN3080
 3060 DC=DC+T:LO=1
 3070 IFDC<=0THENDC=-DC
 3080 POKER+PL,DC
 3090 NEXTPL:RETURN
 4000 REM
 4010 FORPL=0TOSI
 4020 POKEPW+PL,0:POKETE+PL,0:IFPA=1THENPOKER+PL,0
 4050 NEXTPL
 4060 POKEPW,16/PA↑2
 4070 IFPA=1THENDI=5
 4080 IFPA=2THENDI=239
 4090 PT=PW:GOSUB1000:E=1:SN=3-2*PA
 4110 RETURN
 5000 FORPL=0TOSI
 5005 RY=PEEK(PW+PL)
 5010 POKETE+PL,RY
 5030 NEXTPL
 5040 RETURN
READY.
```

Das Basic-Programm zur Berechnung von π jst ein Beispiel dafür, wie man die rechnerinterne Genauigkeitsbeschränkung überwinden sowie nahezu beliebig viele Stellen errechnen und ausgeben kann

Michael Schlierbach

TI-59 errechnet Bundestags-Sitze

Mit programmierbaren Taschenrechnern lassen sich langwierige Rechengänge automatisch erledigen. Das d'Hondtsche Verfahren, nach dem die Bundestagssitze ausgezählt werden, ist dafür ein gutes Beispiel. Bei diesem Programm für den Rechner TI-59 und den Drucker PC-100 C erleichtert die indirekte Adressierung die Programmierarbeit wesentlich.

Zur Verteilung der Sitze im Bundestag wird in der Bundesrepublik das d'Hondtsche Verfahren angewandt. Bei diesem Verfahren werden die Stimmen der einzelnen Parteien nacheinander durch 1; 2; 3;... dividiert, und diejenige Partei, die dabei den höchsten Quotienten hat, bekommt einen Sitz. Ein Beispiel mit den Daten aus der Bundestagswahl 1976 findet man in *Bild 1*.

SPD	214
CDU	190
CSU	53
F. D. P.	39

Bild 1. Zahlenbeispiel für die Bundestagswahlen 1976. Damals waren 496 Mandate für folgende Zweitstimmenzahlen zu vergeben: SPD 16 099 019, CDU 14 367 302, CSU 4 027 499, F.D.P. 2 995 085

Diesen Vorgang kann man mit einem programmierbaren Taschenrechner zu Hause nachvollziehen und sich somit Gewißheit über die Sitze im Bundestag verschaffen.

Das Programm ist so ausgelegt, daß es auf dem Drucker PC 100 C die Parteinamen und die dazugehörigen Sitze in einer Zeile ausdruckt (*Bild 2*). *Tabelle 1* zeigt die Codes für die Parteien. Für

000	76 LBL	044	97 DSZ	088	32 X:T	132	30 30	176	22 INV	216	82 82
001	11 A	045	00 00	089	43 RCL	133	85 +	177	49 PRD	217	43 RCL
002	66 PAU	046	00 00	090	00 00	134	01 1	178	59 59	218	57 57
003	42 STO	047	36 36	091	42 STO	135	00 0	179	00 0	219	52 EE
004	10 10	048	43 RCL	092	40 40	136	95 =	180	42 STO	220	02 2
005	32 X:T	049	10 10	093	61 GTO	137	42 STO	181	57 57	221	22 INV
006	08 8	050	42 STO	094	00 00	138	40 40	182	01 1	222	52 EE
007	00 0	051	00 00	095	77 77	139	73 RC*	183	00 0	223	69 OP
008	42 STO	052	85 +	096	29 CP	140	30 30	184	00 0	224	03 03
009	61 61	053	02 2	097	43 RCL	141	55 ÷	185	49 PRD	225	43 RCL
010	91 R/S	054	00 0	098	40 40	142	73 RC*	186	57 57	226	60 60
011	76 LBL	055	95 =	099	42 STO	143	40 40	187	01 1	227	69 OP
012	12 B	056	42 STO	100	00 00	144	95 =	188	00 0	228	04 04
013	42 STO	057	30 30	101	02 2	145	72 ST*	189	49 PRD	229	01 1
014	20 20	058	01 1	102	00 0	146	00 00	190	59 59	230	22 INV
015	69 OP	059	72 ST*	103	44 SUM	147	61 GTO	191	43 RCL	231	44 SUM
016	20 20	060	30 30	104	00 00	148	00 00	192	59 59	232	61 61
017	43 RCL	061	22 INV	105	01 1	149	68 68	193	59 INT	233	73 RC*
018	00 00	062	44 SUM	106	74 SM*	150	43 RCL	194	22 INV	234	61 61
019	91 R/S	063	30 30	107	00 00	151	10 10	195	44 SUM	235	69 OP
020	76 LBL	064	97 DSZ	108	01 1	152	85 +	196	59 59	236	01 01
021	13 C	065	00 00	109	00 0	153	03 3	197	24 CE	237	01 1
022	72 ST*	066	00 00	110	44 SUM	154	00 0	198	85 +	238	22 INV
023	00 00	067	58 58	111	00 00	155	42 STO	199	53 (	239	44 SUM
024	43 RCL	068	43 RCL	112	01 1	156	30 30	200	24 CE	240	61 61
025	00 00	069	10 10	113	74 SM*	157	95 =	201	75 -	241	73 RC*
026	22 INV	070	42 STO	114	00 00	158	32 X:T	202	06 6	242	61 61
027	67 EQ	071	00 00	115	22 INV	159	01 1	203	93 .	243	69 OP
028	00 00	072	42 STO	116	44 SUM	160	44 SUM	204	05 5	244	02 02
029	15 15	073	40 40	117	20 20	161	30 30	205	54)	245	69 OP
030	85 +	074	73 RC*	118	43 RCL	162	73 RC*	206	69 OP	246	05 05
031	01 1	075	00 00	119	20 20	163	30 30	207	10 10	247	98 ADV
032	00 0	076	32 X:T	120	67 EQ	164	66 PAU	208	85 +	248	43 RCL
033	95 =	077	22 INV	121	01 01	165	42 STO	209	02 2	249	30 30
034	42 STO	078	97 DSZ	122	50 50	166	59 59	210	95 =	250	22 INV
035	30 30	079	00 00	123	43 RCL	167	28 LOG	211	44 SUM	251	67 EQ
036	73 RC*	080	00 00	124	40 40	168	85 +	212	57 57	252	01 01
037	00 00	081	96 96	125	42 STO	169	01 1	213	97 DSZ	253	59 59
038	72 ST*	082	73 RC*	126	00 00	170	95 =	214	58 58	254	25 CLR
039	30 30	083	00 00	127	85 +	171	59 INT	215	01 01	255	91 R/S
040	01 1	084	22 INV	128	01 1	172	42 STO				
041	22 INV	085	77 GE	129	00 0	173	58 58				
042	44 SUM	086	00 00	130	95 =	174	22 INV				
043	30 30	087	77 77	131	42 STO	175	28 LOG				

Bild 2. Das komplette Programm. Sicher finden auch Sie einen Trick, mit dem sich der DSZ-Befehl in Zeile 213–216 eingeben läßt! (Hinweis: Schreiben Sie zuerst RCL statt DSZ und gehen Sie dann auf 213 zurück)

Tabelle 1: Druckcodes der Parteien

Nr.	Partei	Code	Register
1	SPD	0000363316	79
2	CDU	0000151641	77
3	CSU	0000153641	75
4	F.D.P.	0021401640	73
		3340000000	72
5	GRÜNE	2235411731	71
		1700000000	70
6	DKP	0000162633	69
7	NPD	0000313316	67
8	SONSTIGE	3632313637	65
		2422170000	64

Tabelle 2: Bedienung des Rechners

Schritt	Eingabe	Taste	Anzeige
1	Ändern der Speicherbereichsverteilung	CMS 8 OP 17	319.79
2	Einlesen der Magnetkartenseiten 1 und 2	–	2
3	Zahl der Parteien	A	80
4	Zahl der Mandate	B	1
5	Zweitstimmen für i-te Partei	C	i
(Schritt 5 wird solange wiederholt, bis die letzte Stimmenzahl eingegeben ist; dann springt der Rechner selbst ins Hauptprogramm.)			
6	Ausdruck der Ergebnisse	–	Mandate

jede Partei sind zwei Register zum Ablegen des Druckercodes vorgesehen. Der Druck erfolgt in den Druckspalten OP 01 und OP 02.

Die Eingabe der Zweitstimmen muß in der Reihenfolge der in den Registern 79...62 gespeicherten Parteien erfolgen. Der Code der ersten Partei wird dabei in den Registern 79 und 78 gespeichert, der der zweiten in 77 und 76 usw. Mehr als neun Parteien können vom Programm nicht verarbeitet werden, wird dies jedoch nötig, so müßte ein Betriebsprogramm eingebaut werden, das die Daten auf Magnetkarten zwischenspeichert.

Innerhalb des Programms befindet sich eine Routine, die das Ergebnis der Berechnungen in den Druckercode wandelt. Sie kann auch in anderen Programmen verwendet werden und befindet sich auf den Programmspeicherplätzen 165 bis 218. Bei den Daten aus Bild 1 benötigt das Programm ca. $^3/_4$ Stunden. Außer dem Druck erfolgt die Anzeige der Sitze in der richtigen Reihenfolge durch eine Pause auf dem Rechnerdisplay.

Die Bedienungsanleitung des Programms findet man in *Tabelle 2*. Bei der Speicherbereichsverteilung 319,79 (8 OP 17) werden die Druckercodes der Parteien sowie das Register 60 („Sitze") noch auf der Magnetkartenseite 2 mitgespeichert.

Text-Ausdruck mit dem AIM-Drucker

Der AIM-Thermodrucker erlaubt aufgrund seiner Breite von 20 Zeichen zwar eine sehr brauchbare Disassemblierung von Maschinensprache-Programmen, die Wiedergabe von Texten mit größerer Zeilenlänge ist jedoch recht unbefriedigend; speziell bei Basic-Listings läßt die Übersichtlichkeit zu wünschen übrig. Wer sich nur daran stört und etwas mehr Zeitaufwand nicht scheut, muß sich aber deswegen nicht gleich eine TTY zulegen. Mit Hilfe des vorliegenden Programmes kann in mehreren Durchläufen jeweils ein bestimmter Zeilenabschnitt ausgegeben werden. Klebt man die einzelnen Papierstreifen bündig aneinander, erhält man einen Ausdruck voller Breite.

Das Programm wird durch einen Start bei 0FB6 initialisiert. Der DILINK-Vektor (A406) zeigt dann auf den Beginn der eigentlichen Routine bei 0FC1. Vor jedem Druckdurchgang sind die Werte für die Zeropage-Adressen FE und FF entsprechend der *Tabelle* einzusetzen, die den auszudruckenden Zeilensektor bestimmen. Da – um die Monitorroutine mitverwenden zu können – nur jeweils 19 Zeichen gedruckt werden, sind dies im ersten Durchlauf die Stellen 0 bis 18, im zweiten 19 bis 37 usw. Die Änderung kann bei Basic-Programm-Ausdrucken mit den angegebenen Poke-Befehlen erfolgen. Die Zahl der erforderlichen Druckstreifen hängt natürlich von der längsten vorkommenden Zeile ab. Solange die Routine angesprungen wird, erscheinen auf dem Display stets nur die ersten 20 Zeichen einer Zeile, so daß bei jedem Ausdrukken die Basic-Zeilennummern angezeigt werden. Die Zahl der Durchläufe sollte daher vor der Initialisierung bestimmt werden.

Zum Schluß ist der DILINK-Vektor wieder zurückzusetzen. Am einfachsten geschieht dies durch eine beliebige Änderung des NMI-Vektors bei A402/A403. Betätigt man anschließend die Reset-Taste, so erfolgt ein „Kalter Reset", bei dem u. a. auch DILINK wieder auf EF05 gerichtet wird.

Herbert Müller

```
OFB6 A9 LDA #C1
OFB8 8D STA A406
OFBB A9 LDA #0F
OFBD 8D STA A407
OFC0 00 BRK
OFC1 48 PHA
OFC2 29 AND #7F
OFC4 C9 CMP #0D
OFC6 D0 BNE OFCC
OFC8 68 PLA
OFC9 4C JMP EF05
OFCC 68 PLA
OFCD 48 PHA
OFCE 20 JSR EB9E
OFD1 A8 TAY
OFD2 AE LDX A415
OFD5 E4 CPX FE
OFD7 90 BCC OFE8
OFD9 E4 CPX FF
OFDB B0 BCS OFE8
OFDD 8A TXA
OFDE 38 SEC
OFDF E5 SBC FE
OFE1 8D STA A416
OFE4 98 TYA
OFE5 20 JSR F000
OFE8 EE INC A415
OFEB E0 CPX #14
OFED B0 BCS OFF6
OFEF CE DEC A415
OFF2 98 TYA
OFF3 20 JSR EF05
OFF6 A9 LDA #13
OFF8 8D STA A416
OFFB 20 JSR EBAC
OFFE 68 PLA
OFFF 60 RTS
```

Dieses AIM-65/PC-100-Programm ermöglicht, durch Aneinanderkleben mehrerer Druckstreifen einen Ausdruck voller Zeilenbreite zu erzielen

Tabelle: Parameter in der Zero Page

Durchgang	Hex 00FE	Hex 00FF	Basic Poke 254,	Basic :Poke 255,
1	00	13	0	19
2	13	26	19	38
3	26	39	38	57
4	39	4C	57	76

Rudolf Hofer

BASIC vor dem Handbuch

3. Teil

In unserem BASIC-Grundkurs geht es hier um das Verarbeiten von Tabellen mit dem READ-DATA-Mechanismus. Die Ausführungen beziehen sich zwar im Prinzip auf den TRS-80, sind aber ohne weiteres auf andere Computer übertragbar.

Noch ein Wort zum Vertauschen der Variablen-Werte. Dieser Vorgang spielt sich in den Zeilen 210, 220 und 230 (*Bild 8*) ab. Der Wert von A$(I) wird zunächst mit Hilfe der Variablen H$ zwischengespeichert. A$(I) kann dann den Wert von A$(I+K) übernehmen. Der zwischengespeicherte Wert wird jetzt A$(I+K) zugewiesen. H$ fungierte nur als Hilfsvariable. Ihr Wert wird im weiteren Verlauf des Programmes nicht mehr benötigt.

2. Schritt: I=2, K=1, R=2

A$(2)<A$(3)?, nein weil DORA>BERTA
A$(2)=BERTA, A$(3)=DORA
K=R?, nein
K=K+1=2
A$(2)<A$(4)?, ja weil BERTA<CAESAR
K=R?, ja
R=R-1=1
I=N-1?, nein

3. Schritt: I=3, K=1, R=1
A$(3)<A$(4)?, nein weil DORA>CAESAR
A$(3)=CAESAR, A$(4)=DORA
K=R?, ja
I=N-1, ja
Programmende

Als Ergebnis erhalten wir die Namen in der gewünschten Reihenfolge, d.h. die Variablen mit einem niedrigeren Index haben einen kleineren Wert. Um die sortierte Liste ausdrukken zu lassen genügt, eine einfache FOR-NEXT-Schleife.

Bild 8 zeigt das gesamte Programm. Man sieht, daß mit Hilfe von FOR-NEXT-Schleifen die Sache sehr einfach wird, da hierbei die Indizes I und K automatisch die gewünschten Werte bis zu den Grenzen N-1 und R durchlaufen. Es empfiehlt sich, das Beispiel noch einmal in allen Schritten durchzuspielen und dabei mit dem BASIC-Programm zu vergleichen, was passiert. Wie gesagt: Wenn Sie dieses Problem durchschauen, wird Ihr weiterer Weg leicht zu gehen sein.

Noch eine Bemerkung zu indizierten Variablen: Bei manchen BASIC-Versionen ist der höchste zulässige Index 255, bei anderen ist er niedriger. Index Null ist nicht immer zugelassen. Da für jede Variable im Speicher Platz reserviert wird, setzt der Computer nach dem Einschalten aber einen kleineren Wert (z.B. 10) für den höchsten Index fest. Solange der Index darunter bleibt, stört uns das nicht. Wenn wir aber größere Indizes brauchen (etwa 20 Namen sortieren wollen), müssen wir das der Maschine mitteilen. Wir tun das mit dem DIM-Befehl. Soll der größte Index der Variablen A$(I) beispielsweise 20 sein, dann müssen wir den Befehl

DIM A$(20)

einfügen, bevor wir mit A$(I) zu arbeiten beginnen. Ebenso könnten wir aber

DIM A$(N)

verwenden, wobei N vom Programm ein Wert (z.B. mit INPUT) zugewiesen wird.

Es ist auf alle Fälle empfehlenswert, nicht zu großzügig zu „dimensionieren", da hierfür Speicherplatz reserviert wird, der für andere Zwecke verlorengeht.

```
7o CLEAR 1ooo
8o PRINT "WIEVIELE NAMEN";
9o INPUT N
1oo FOR I=1 TO N
11o INPUT A$(I)
12o NEXT I
17o R=N-1
18o FOR I=1TO N-1
19o FOR K=1 TO R
2oo IF A$(I)<A$(I+K) THEN 24o
21o H$=A$(I)
22o A$(I)=A$(I+K)
23o A$(I+K)=H$
24o NEXT K
25o R=R-1
26o NEXT I
3oo FOR I=1 TO N
31o PRINT A$(I)
32o NEXT I

WIEVIELE NAMEN? 5
? BERTA
? THEODOR
? ANTON
? CHARLIE
? CHARLY
ANTON
BERTA
CHARLIE
CHARLY
THEODOR
```

Bild 8. Sortierprogramm und Anwendungsbeispiel einer Tabelle

Wir verarbeiten Tabellen

Indizierte Variablen werden auch als Felder bezeichnet. Ist ein einziger Index vorhanden, spricht man von einem eindimensionalen Feld (oder Array). Moderne BASIC-Computer lassen aber auch zwei- oder gar dreidimensionale Felder zu. Damit lassen sich besonders einfach Tabellen verarbeiten.

Beispiel: Es soll eine Tabelle angelegt werden, die zeilenweise nach Schülernamen und spaltenweise nach Fächern aufgeteilt ist. Am jeweiligen Schnittpunkt steht die Note, die der Schüler im entsprechenden Fach hat (*Bild 9*). Sie soll einem Element des zweidimensionalen Feldes N(Z,S) zugeordnet werden, wobei Z die Zeilennummer und S die Spaltennummer darstellt.

Das Programm soll die Möglichkeit bieten, die Durchschnittsnote jedes Schülers und die Durchschnittsnote jedes Faches auszurechnen.

Für das Programm benötigen wir drei Felder: S$(1)...S$(6) für die Schülernamen, F$(1)...F$(4) für die Fächer und N(1,1)...N(6,4) für die Noten. Bitte beachten Sie, daß man für die Schülernamen und Fächer String-Variablen (gekennzeichnet durch das $-Zeichen) benötigt, während die Noten durch numerische Variablen ausgedrückt werden, da man mit ihnen ja rechnen will.

Bild 10 zeigt noch einmal die gewünschte Anordnung mit den Variablen-Namen.

Um die Tabelle einzulesen, müssen wir also jeden Wert in Bild 9 der entsprechenden Variablen in Bild 10 zuweisen. Am einfachsten macht man das mit den Befehlen READ und DATA.

Mit folgendem Programmteil liest man z.B. das Feld S$(I) ein. I wurde hier willkürlich gewählt und stellt lediglich den veränderlichen Index dar. Genauso gut könnte man jeden anderen Buchstaben verwenden.

```
210 FOR I = 1 TO 6
220 READ S$(I)
230 NEXT I
1000 DATA AST, BAUM, DORN, FAUST, GROB, ZAUN
```

Wir sehen, daß innerhalb der FOR-NEXT-Schleife der Reihe nach die Befehle READ S$(1), READ S$(2)...READ S$(6) ausgeführt werden. Das veranlaßt den Computer, nach der ersten Zeile im Programm zu suchen, die mit DATA beginnt. Die Werte (in diesem Fall Namen) in dieser Zeile (durch Komma getrennt) werden dann der Reihe nach den Variablen S$(1)...S$(6) zugewiesen. Damit erreichen wir dasselbe wie mit einem INPUT-Befehl, allerdings muß der Benutzer während der Programmausführung keine Eingabe mehr machen.

DATA-Zeilen werden üblicherweise am Programmende angefügt. Zu beachten sind zwei Dinge: Erstens muß der Variablen-Typ im READ-Befehl mit dem im DATA-Befehl übereinstimmen. Es darf also nicht eine numerische Variable im READ-Befehl stehen, während in der dazugehörigen DATA-Zeile Strings stehen. Zweitens sucht der Computer automatisch nach der nächsten DATA-Zeile, wenn er in der vorherigen keine Variable mehr findet.

Der Computer merkt sich während des gesamten Programmdurchlaufs, welche DATA-Werte er bereits gelesen hat, und macht bei jedem weiteren READ-Befehl an der folgenden Stelle weiter. Nach dem Befehl

```
RESTORE
```

fängt er wieder bei der ersen DATA-Zeile an.

Zurück zu unserem Beispiel: Genauso wie das Feld S$(I) wird das Feld F$(I) eingelesen:

```
250 FOR J = 1 TO 4
260 READ F$(J)
270 NEXT J
1010 DATA MATHEMATIK, PHYSIK, DEUTSCH,
ENGLISCH
```

Wieder spielt es keine Rolle, welchen Buchstaben wir als Index verwenden. Das wird erst aktuell, wenn wir z.B. ein mehrdimensionales Feld einlesen, wie im folgenden. Nun wollen wir noch das Feld N(Z,S) einlesen. Das geht in folgenden Schritten vor sich:

1. Wir setzen die Zeilennummer Z auf 1 und lesen N(1,1)...N(1,4) ein. Am leichtesten geht das wieder mit einer FOR-NEXT-Schleife.
2. Wir erhöhen die Zeilennummer auf 2 und lesen N(2,1)...N(2,4) ein.

Diesen Vorgang wiederholen wir so lange, bis die höchste Zeilennummer erreicht ist.

Das Programm (*Bild 11*) besteht aus zwei ineinandergeschachtelten FOR-NEXT-Schleifen, von denen die innere bei jedem Durchlauf der äußeren viermal durchlaufen wird. Geht man die Befehlsfolge der Reihe nach durch, dann sieht man sofort, daß die Werte von Bild 9 zeilenweise auf die Variablen von Bild 10 übertragen werden.

Um die Übersichtlichkeit zu wahren, wurden genau sechs DATA-Zeilen, entsprechend sechs Schülern, angelegt. Man könnte aber auch alle Werte in eine Zeile fortlaufend eintip-

FACH / NAME	Math.	Phys.	Deutsch	Englisch
Ast	2	3	3	1
Baum	4	3	4	4
Dorn	1	2	1	3
Faust	3	3	1	1
Grob	5	4	6	4
Zaun	2	2	2	1

Bild 9. Beispiel einer Tabelle

	F$ (1)	F$ (2)	F$ (3)	F$ (4)
S$(1)	N (1,1)	N (1,2)	N (1,3)	N (1,4)
S$(2)	N (2,1)	N (2,2)	N (2,3)	N (2,4)
S$(3)	N (3,1)	N (3,2)	N (3,3)	N (3,4)
S$(4)	N (4,1)	N (4,2)	N (4,3)	N (4,4)
S$(5)	N (5,1)	N (5,2)	N (5,3)	N (5,4)
S$(6)	N (6,1)	N (6,2)	N (6,3)	N (6,4)

Bild 10. Zuordnung der Tabellenelemente aus Bild 9 zu Elementen der Felder S$ (Z), F$ (S) und N (Z, S)

```
280 FOR Z=1 TO 6
290 FOR S=1 TO 4
300 READ N(Z,S)
310 NEXT S
320 NEXT Z

1020 DATA 2,3,3,1
1030 DATA 4,3,4,4
1040 DATA 1,2,1,3
1050 DATA 3,3,1,1
1060 DATA 5,4,6,4
1070 DATA 2,2,2,1
```

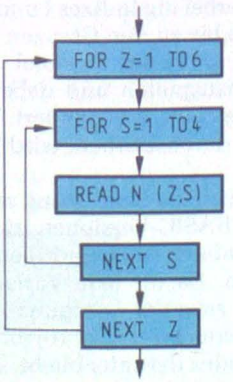


Bild 11. Einlesen eines zweidimensionalen Feldes

pen. Allerdings können Änderungen dann nicht ganz so bequem durchgeführt werden, wie dies im gewählten Beispiel der Fall ist.

Die gestellte Aufgabe lautete: Es sind die Durchschnittsnoten der Schüler und Fächer zu berechnen. Bevor wir das gesamte lauffähige Programm dafür behandeln, wollen wir das Prinzip dieser Berechnung diskutieren. Welche Kriterien stehen uns dafür zur Verfügung?

- Die Namen der Schüler liegen als Feld S$(1)...S$(6) vor
- Die Fächer liegen als Feld F$(1)...F$(4) vor.
- Die Noten liegen als Feld N(1,1)...N(6,4) vor. Der erste Index entspricht dabei dem Schüler-Index, der zweite dem Fach-Index.

Das letzte Kriterium birgt die Lösung des Problems. Wollen wir z. B. die Durchschnittsnote des Schülers DORN (S$(3)) berechnen, dann müssen wir alle Noten addieren, deren erster Index 3 ist, und durch die Fächerzahl teilen. Wollen wir aber etwa die Durchschnittsnote des Faches MATHEMATIK wissen, dann ist die Summe der Noten mit dem zweiten Index 1 durch die Zahl der Schüler zu teilen.

Bild 12 zeigt einen Programmteil, mit dem man z.B. den Durchschnitt eines Schülers berechnen kann.

In Zeile 610 wird vom Benutzer der Zeilen-Index Z erfragt, der angibt, um welchen Schüler es sich handelt (z.B 1 ≙ AST). Innerhalb der FOR-NEXT-Schleife werden dann die Noten der einzelnen Fächer aufaddiert. Die Gesamtsumme stellt die Variable G dar, die vor der Schleife sicherheitshalber auf Null gesetzt wird. Verfolgt man den Ablauf, dann stellt man fest, daß beim ersten Schleifendurchlauf G = 0+N(1,1) wird (vorausgesetzt für Z wurde 1 gewählt). Beim zweiten Durchlauf wird zum gegenwärtigen Wert von G, nämlich N(1,1), der Wert von N(1,2) hinzugezählt. Nach vier Durchläufen ist G die Summe aller Fächernoten des Schülers; in Zeile 660 wird direkt im PRINT-Befehl der Durchschnitt G/4 berechnet. Z hat immer noch den Wert, der dem Schüler entspricht. Das wird ausgenutzt, und zwar, um den Namen des Schülers S$(Z) auszudrucken.

Analog dazu wird in *Bild 13* der Durchschnitt eines Faches berechnet und ausgedruckt. Natürlich können diese Teilprogramme nur dann funktionieren, wenn die entsprechenden Felder vorher wie beschrieben eingelesen wurden. Versuchen Sie doch, aus den einzelnen Teilen ein Gesamtprogramm zu machen. Hierzu einige Hinweise:

Am Anfang des Programmes müssen Sie die Felder dimensionieren und Speicherplatz für String-Variable reservieren, z.B.

```
DIM N(6,4)
DIM S$(6)
DIM F$(4)
CLEAR 1000
```

bei TRS-80 (bei anderen Maschinen im Handbuch nachsehen, bei PET/CBM überflüssig).

Die DIM-Befehle müssen natürlich entsprechend abgewandelt werden, wenn man mehr Namen und Fächer zulassen will. CLEAR 1000 bedeutet, es wird Speicher für 1000 String-Zeichen freigehalten. Beim PET 2001 kann dieser Befehl entfallen.

Es folgt die Eingabe der Felder. Die zugehörigen DATA-Zeilen stellt man an das Ende des Programms. Nun muß der Benutzer die Gelegenheit erhalten, zu sagen, was er will: den Durchschnitt eines Schülers oder eines Faches. Das geht z.B. mit dem INPUT-Befehl, gefolgt von einer IF-THEN-Entscheidung, die je nach Eingabe des Benutzers zum entsprechenden Programmteil verzweigt.

In den Bildern 12 und 13 wird der Index Z bzw. S als Zahl erfragt. Eleganter wäre es, der Benutzer könnte einfach den Namen des Schülers oder des Faches eingeben und würde ohne weitere Umwege den gewünschten Durchschnitt erfahren. In diesem Fall müßte der Index nicht per INPUT erfragt, sondern vom Programm ermittelt werden. Man könnte das so lösen:

```
370 INPUT B$
380 FOR Z = 1 TO 6
390 IF S$(Z) = B$ THEN 600
400 NEXT Z
```

Gibt der Benutzer einen Namen ein, der im Feld enthalten ist, dann fällt der Vergleich in Zeile 390 irgendwann positiv aus. Z hat zu diesem Zeitpunkt den Wert, der dem Schüler entspricht. Läßt man in Bild 12 die Zeile 610 weg, dann wird der Durchschnitt des Schülers errechnet, dessen Name in Zeile 370 eingegeben wurde.

Wendet man diese Methode an, dann kann man sogar auf die Abfrage verzichten, was der Benutzer will. Wird nämlich kein Name, sondern ein Fach in Zeile 370 eingetippt, verläßt das Programm die Schleife auf dem „normalen" Weg und macht bei dem Befehl weiter, der nach Zeile 400 erscheint. Hier kann man in gleicher Weise die Fächer abfragen:

```
450 FOR S = 1 TO 4
460 IF F$(S) = B$ THEN 700
470 NEXT S
```

Natürlich muß dann in Bild 13 die Zeile 710 gestrichen werden. Der Rest, um das Programm zu optimieren, besteht darin, den weiteren Ablauf festzulegen, nachdem eine Berechnung durchgeführt wurde oder kein Name bzw. Fach gefunden wurde, das dem eingegebenen Wort entspricht. Eine Möglichkeit wäre, das Programm als endlose Schleife auszulegen, also nach jeder Berechnung wieder auf den INPUT-Befehl zu springen, der vom Benutzer die Eingabe eines Wortes verlangt. *(Schluß folgt)*

```
600 G=0
610 INPUT Z
620 FOR S=1 TO 4
630 G=G+N(Z,S)
640 NEXT S
650 PRINT "DER SCHUELER ";S$(Z);" HAT EINEN"
660 PRINT "NOTENDURCHSCHNITT VON ";G/4
```

Bild 12. Programmteil zur Berechnung des Schüler-Notendurchschnitts

```
700 E=0
710 INPUT S
720 FOR Z=1 TO 6
730 E=E+N(Z,S)
740 NEXT Z
750 PRINT "DIE DURCHSCHNITTSNOTE IM FACH "
760 PRINT F$(S); " IST ";E/6
```

Bild 13. Programmteil zur Berechnung des Fach-Notendurchschnitts

Christian Buess

Plotten mit dem CBM 3022

Der CBM 3022-Drucker (Tractor feed) bietet einige sehr interessante Möglichkeiten, die für das Plotten von Funktionen äußerst nutzvoll sind, z. B. variabler Zeilenabstand (beliebig programmierbar) und Definition eigener Zeichen, d. h. jeder Matrixpunkt ist getrennt ansteuerbar.

Setzt man diese Möglichkeiten optimal ein, so kann man auf einem A4-Blatt 480 x 720 = 345600 Punkte ansteuern. Das Ganze kompliziert sich noch etwas durch die Tatsache, daß in einer (Ausgabe-)Zeile nur ein Sonderzeichen stehen darf; man muß also einzelne Zeilen mehrfach durchlaufen.

Das Programm in *Bild 1* arbeitet nun mit einer solchen maximalen Auflösung (0,4 mm in beide Richtungen).
Es stehen 3 Möglichkeiten zur Auswahl:
Nur Plotten:
Funktionsplott auf volle Breite, d. h. 80 Zeichen = 480 Punkte pro Linie.
Plotten und X-Wert:
Funktionsplott auf die Breite von 70 Zeichen = 420 Punkte pro Linie, dazu noch der X-Wert auf 3 Stellen + Exp. (x-Achse hier vertikal).
Plotten und X-, Y-Wert:
Funktionsplott auf die Breite von 50 Zeichen = 300 Punkte pro Linie, X-Wert, Y-Wert mit 9 Stellen + Exp. (Y-Wert der Hauptfunktion).

Das Programm ist so ausgelegt, daß mehrere Funktionen gleichzeitig ausgedruckt werden können. In der vorliegenden Version sind das maximal 5 Kurven (auf demselben Intervall). Eine Abänderung für mehr Funktionen ist aber sehr leicht möglich.

Zu jeder Funktion muß auch noch angegeben werden, in welcher von 4 Stricharten sie geplottet werden soll. Dabei stehen folgende Möglichkeiten zur Auswahl: durchgezogen, Striche, Striche/Punkte, oder nur Punkte (*Bild 2*).

Viele Funktionen sind nicht über das ganze Intervall stetig; es besteht aber oft der Wunsch, Funktionen über solche Unstetigkeitsstellen hinaus zu plotten. Beispiel:
$y = 1/x$ ist nicht definiert im Punkt $x = 0$.

Ein normales Plottprogramm würde beim Punkt Null mit der Fehlermeldung „division by zero" aussteigen. Beim folgenden Programm können solche Punkte vorher eingegeben werden, und diese werden dann übersprungen. Eingegeben wird die Anzahl der Unstetigkeitsstellen n und, bei $n \neq 0$, noch 5 Zahlen, wobei die ersten n als Unstetigkeitsstellen gerechnet werden (die Zahl „5" kann leicht erhöht werden).

Bei der Eingabe der Funktionen wurde noch ein kleiner Trick verwendet, der eine „Selbstprogrammierung" des Computers bewirkt: Die n Funktio-

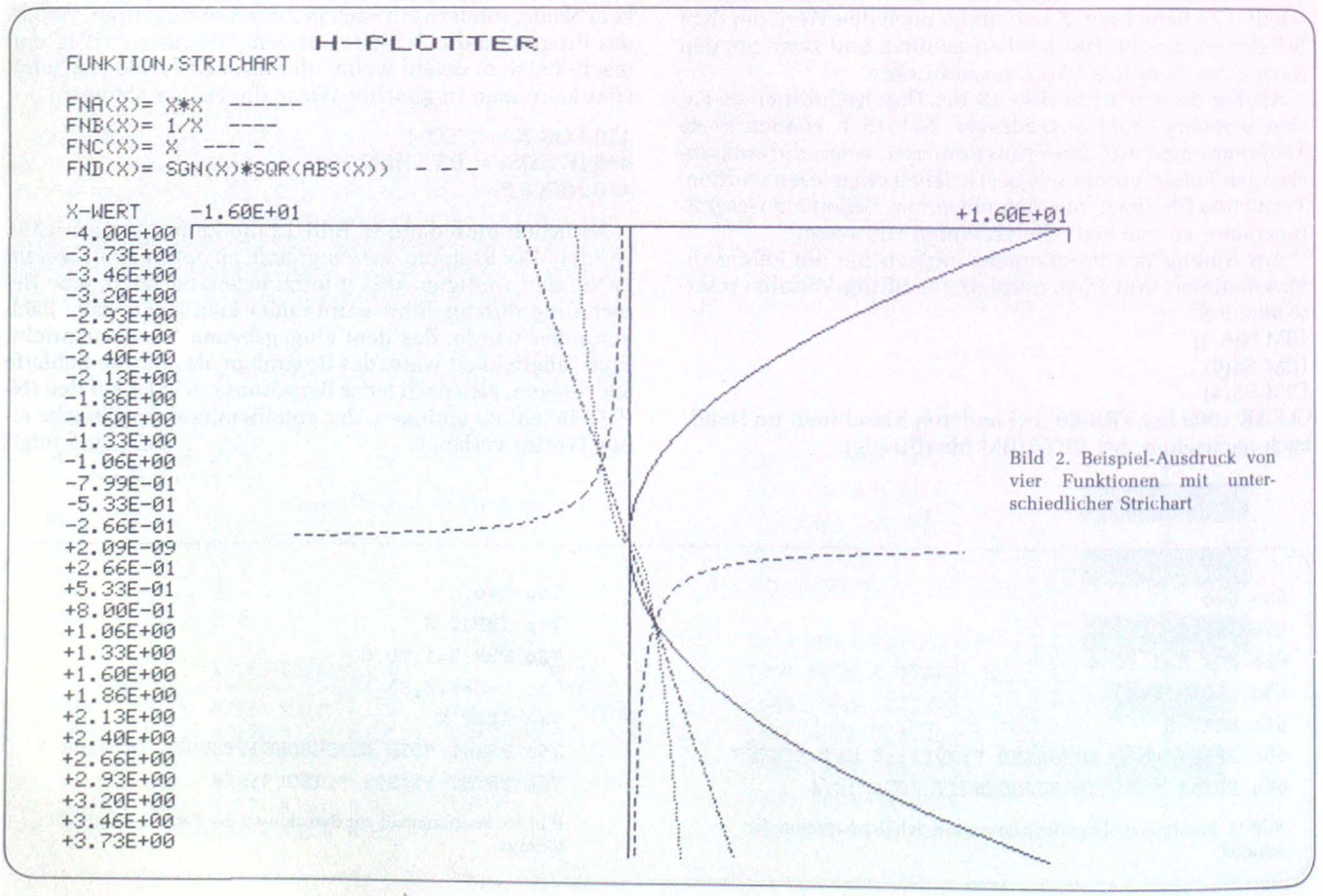


Bild 2. Beispiel-Ausdruck von vier Funktionen mit unterschiedlicher Strichart

```
100 REM*--------------------------------*
110 REM H-PLOTTER                CH.BUESS
120 REM =========                11.5.80
130 REM MIT DEM FOLGENDEN PROGRAMM
140 REM KOENNEN BIS ZU 5 KURVEN MIT 4
150 REM BELIEBIGEN STRICHARTEN GE-
160 REM ZEICHNET WERDEN.
170 REM AUFLOESUNG: 0.4MM !!
180 REM*--------------------------------*
190 REM
200 REM ------------------------------
210 REM  FUNKTION-,STRICHART-EINGABE
220 REM ------------------------------
230 REM
240 DATA0,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,0,1,0,1,0,1,0,1,0
250 PRINT"                    H-PLOTTER"
260 PRINT"                                                "
270 PRINT" MIT DIESEM PROGAMM KOENNEN BIS"
280 PRINT" FUENF FUNKTIONEN AUFGEZEICHNET WERDEN."
290 PRINT" FOLGENDE STRICHARTEN STEHEN ZUR"
300 PRINT" VERFUEGUNG :"
310 PRINT"    ------------  (1)"
320 PRINT"    ----  ----    (2)"
330 PRINT"    --- - --- -   (3)"
340 PRINT"    - - - - - -   (4)"
350 INPUT" ANZAHL FUNKTIONEN ";C
360 PRINT" EINGABE: STRICHART,FUNKTION"
370 DIMA$(C),K(6*C):N=0
380 FORI=1TOC
390 INPUT"                    ";S,A$(I)
400 FORJ=0TO6*(S-1):READK:NEXT
410 FORJ=0TO5:READK:K(N*6+J)=K:NEXT
420 PRINT"                    "
430 PRINT"                                                "
440 N=N+1:RESTORE:NEXT
450 PRINT":OPEN1,4
460 PRINT#1,CHR$(1);"          H-PLOTTER":PRINT#1,
    "FUNKTION,STRICHART":PRINT#1
470 FORI=1TOC
480 PRINT669+I;"DEFFN";CHR$(64+I);"(X)";"=";A$(I)
490 PRINT#1,"FN";CHR$(64+I);"(X)= ";A$(I);"  ";
500 FORJ=0TO5:PRINT#1,CHR$(32+13*K((I-1)*6+J));:NEXT:PRINT#1
510 NEXT:PRINT#1
520 PRINT670+C;"AK=";C
530 PRINT671+C;"DATA";
540 FORI=0TO6*C-1:IFK(I)=1THENPRINT"1,";
550 IFK(I)=0THENPRINT"0,";
560 NEXT
570 PRINT" ":PRINT"RUN670"
580 POKE623,19
590 FORI=1TOC+4:POKE623+I,13:NEXT
600 POKE158,C+5
610 END
620 REM
630 REM ---------------
640 REM  DATEN-EINGABE
650 REM ---------------
660 REM
680 PRINT"  DATENEINGABE !"
690 PRINT"                "
700 PRINT"WELCHE DARSTELLUNG: - NUR PLOT     (1)"
710 PRINTTAB(20);"- X UND PLOT   (2)"
720 PRINTTAB(20);"- X,Y UND PLOT (3)"
730 INPUTC:M=78:L$="":IFC=2THENM=68:L$="          "
740 IFC=3THENM=49:L$="                "
750 DIMA%(M,6),K(5,5),P1(AK),Q1(AK),C(AK),U(AK),W(AK,5)
760 LO=LOG(10)
770 RESTORE:FORI=1TO25:READS:NEXT
780 FORJ=1TOAK:FORI=0TO5:READS:K(J,I)=S:NEXTI,J
790 INPUT"ANFANGS-, ENDWERT X ";XA,XB
800 INPUT"DELTA X             ";DX
810 INPUT"Y(MAX), Y(MIN)      ";YA,YB
820 PRINT"EINGABE NICHT DEFINIERTER PUNKTE:":PRINT" 1. ANZAHL"
830 PRINT" 2. WENN (1)<>0 DANN 5 EINGABEN (GE-"
840 PRINT"    ORDNET), FALLS WENIGER ALS 5, DANN"
850 PRINT"    RESTLICHE EINGABEN =0."
860 PRINT"ANZAHL  --------5 EINGABEN--------"
870 FORI=1TOAK:INPUT";W(I,0)
880 IFW(I,0)<>0THENINPUT";W(I,1),W(I,2),W(I,3),W(I,4),W(I,5)
890 PRINT";                                              ":NEXT
900 OPEN1,4,1:OPEN2,4,2:OPEN5,4,5:OPEN6,4,6:OPEN9,4:PRINT#6,CHR$(24)
910 PRINT"BREAK: TASTE S !!":PRINT"AM SCHLUSS : DELETE 670-679"
920 REM
930 REM ===============
940 REM  HAUPTPROGRAMM
950 REM ===============
960 REM
970 IFM=78THENGOSUB1640
980 IFM=49THENGOSUB1750
990 IFM=68THENGOSUB1880
1000 NP=-M*YA/(YB-YA):NQ=6*(NP-INT(NP)):D=1
1010 NL=0:IFNP<0ORNP>=79THENNL=1
1020 FORI=1TOAK:C(I)=0:NEXT
1030 FORX=XATOXBSTEPDX
1040 PRINT#6,CHR$(16):NS=127:IFPEEK(151)=40ORD=99GOTO1100
1050 FORJ=1TOAK:H%=640:FORI=0TO6
1060 GOSUB1160:NEXT:NEXT:GOSUB1320
1070 PRINT#6,CHR$(7):NS=7
1080 FORJ=1TOAK:H%=40:FORI=7TO10
1090 GOSUB1160:NEXT:NEXT:GOSUB1320:NEXT:D=99:GOTO1040
1100 PRINT#1,CHR$(141);:PRINT#6,CHR$(24):PRINT#1:END
1110 REM
1120 REM -------------------------
1130 REM  BERECHNUNG EINER ZEILE
1140 REM -------------------------
1150 REM
1160 HX=X+I/10*DX:IFW(J,0)=0GOTO1180
1170 IFHX=W(J,W(J,0))THENW(J,0)=W(J,0)-1+INT(I/10):C(J)=0:RETURN
1180 ONJGOSUB2090,2100,2110,2120,2130
1190 P=M*(Y-YA)/(YB-YA):Q=6*(P-INT(P))+1:IFP<0ORP>MTHENC(J)=0:GOTO1260
1200 IFC(J)=0THENP1(J)=P:C(J)=1:Q1(J)=Q
1210 P%=ABS(INT(P1(J))*6+INT(Q1(J))-INT(P)*6-INT(Q)):IFP%>1THENGOSUB1490
1220 IFI<>10THENP1(J)=P:Q1(J)=Q
1230 A%(P,Q)=A%(P,Q)OR(H%/10*K(J,U(J)))
1240 IFH%>=10ANDH%<=640THENA%(P,0)=1:U(J)=U(J)+1
1250 IFU(J)>5THENU(J)=0
1260 H%=H%/2:RETURN
1270 REM
1280 REM ------------------------
1290 REM  AUSDRUCK EINER ZEILE
1300 REM ------------------------
1310 REM
1320 L=0:IFNL=0THENA%(NP,NQ+1)=NS:A%(NP,0)=1
1330 FORS=0TOM
1340 IFA%(S,0)=0GOTO1400
1350 A$="":IFL=1THENPRINT#9,CHR$(141);
```

nen werden zuerst per Input in A$(n) abgespeichert, zusammen mit der jeweiligen Strichart. Der Bildschirm wird daraufhin gelöscht, und es werden einige Zeilen Programm vom Computer auf das Display gebracht, abgeschlossen mit RUN 670. Das eigentliche Eingabeprogramm endet bei 610, vorher werden aber noch so viele „carriage return" wie nötig in den Eingabepuffer gespeichert. Die ausgedruckten Zeilen werden gespeichert, und es erfolgt ein neuer Start bei 670. Ein langwieriges Einprogrammieren der Funktionen ist auf diese Weise überflüssig.

```
1360 FORN=1TO6:A$=A$+CHR$(A%(S,N)):A%(S,N)=0:NEXT
1370 A%(S,0)=0:PRINT#5,A$:PRINT#9,L$;
1380 FORH=0TOINT(S):PRINT#9," ";:NEXT
1390 PRINT#9,CHR$(254);:L=1
1400 NEXT:IFI>9GOTO1430
1410 IFM<=68THENZ=X:PRINT#9,CHR$(141);:GOSUB2010:PRINT#1,Z,E;
1420 IFM=49THENZ=FNA(X):GOSUB2010:PRINT#1,Z,E;
1430 PRINT#9,CHR$(13);:RETURN
1440 REM
1450 REM ------------------------------
1460 REM  INTERPOLATION ZW. 2 PUNKTEN
1470 REM ------------------------------
1480 REM
1490 HQ=INT(Q1(J)):L=0:H%=H%*2:HP=INT(P1(J)):V=1:IFP<P1(J)THENV=-1
1500 FORS=P%-1TO1STEP-1
1510 IFS/P%<=.5ANDL=0THENH%=H%/2:L=1
1520 HQ=HQ+V
1530 IFHQ>6THENHQ=1:HP=HP+1
1540 IFHQ<1THENHQ=6:HP=HP-1
1550 A%(HP,HQ)=A%(HP,HQ)OR(H%/10*K(J,U(J)))
1560 IFH%>=10ANDH%<=640THENA%(HP,0)=1:U(J)=U(J)+1
1570 IFU(J)>5THENU(J)=0
1580 NEXT:RETURN
1590 REM
1600 REM --------------------------
1610 REM  PLOTKOPF FUER "NUR PLOT"
1620 REM --------------------------
1630 REM
1640 PRINT#2,"SZ.99■ESZZ                                    ";
1650 PRINT#2,"                                  SZ.99■ESZZ"
1660 Z=YA:GOSUB2010:PRINT#1,Z,E;:Z=YB:GOSUB2010:PRINT#1,Z,E
1670 PRINT#9,"┌──────────────────────────────────────────";
1680 PRINT#9,"──────────────────────────────────────┘";CHR$(141);
1690 RETURN
1700 REM
1710 REM --------------------------
1720 REM  PLOTKOPF FUER "X UND PLOT"
1730 REM --------------------------
1740 REM
1750 PRINT#2,"AAAAAA      AAAAAA            SZ.99■ESZZ ";
1760 PRINT#2,"                                SZ.99■ESZZ"
1770 PRINT#1,"X-WERT",CHR$(29),"Y=F(X)",CHR$(29);
1780 Z=YA:GOSUB2010:PRINT#1,Z,E;:Z=YB:GOSUB2010:PRINT#1,Z,E;
1790 PRINT#9,"                                ┌──────";
1800 PRINT#9,"──────────────────┘";CHR$(141);
1810 PRINT#2,"SZ.99■ESZZ   SZ.999999■ESZZ"
1820 RETURN
1830 REM
1840 REM --------------------------
1850 REM  PLOTKOPF FUER "X,Y UND PLOT
1860 REM --------------------------
1870 REM
1880 PRINT#2,"AAAAAA    SZ.99■ESZZ                    ";
1890 PRINT#2,"                                SZ.99■ESZZ"
1900 PRINT#1,"X-WERT",CHR$(29);
1910 Z=YA:GOSUB2010:PRINT#1,Z,E;:Z=YB:GOSUB2010:PRINT#1,Z,E;
1920 PRINT#9,"          ┌──────────────────────────────";
1930 PRINT#9,"──────────────────────────────────┘";CHR$(141);
1940 PRINT#2,"SZ.99■ESZZ "
1950 RETURN
1960 REM
1970 REM ---------------------
1980 REM  UP FUER EXP. AUSGABE
1990 REM ---------------------
2000 REM
2010 IFZ=0THENE=0:GOTO2030
2020 E=INT(LOG(ABS(Z))/LO):Z=Z/10↑E
2030 RETURN
2040 REM
2050 REM -----------------
2060 REM  FUNKTIONSAUFRUF
2070 REM -----------------
2080 REM
2090 Y=FNA(HX):RETURN
2100 Y=FNB(HX):RETURN
2110 Y=FNC(HX):RETURN
2120 Y=FND(HX):RETURN
2130 Y=FNE(HX):RETURN
```

Bild 1. Basic-Programm zum Plotten auf dem Drucker CBM 3022

Enträtselung römischer Zahlen

Es mag verwundern, wenn auf den Mikrocomputerseiten einer Zeitschrift ein ausgestorbenes Zahlensystem in Erinnerung gebracht wird, doch bietet die Vorschrift zur Umwandlung in heute gebräuchliche Zahlen einen schönen Algorithmus an, der einmal in einer Programmiersprache formuliert werden sollte.

Zudem sei festgestellt, daß außer für die Monatsbezeichnungen im Datum ganz feiner Leute die römischen Zahlenzeichen kaum verwendet werden,

```
110 INPUT"ROEM. ZAHLEN";R$
120 L=LEN(R$)
130 FORI=1TOL
140 A$=MID$(R$,I,1)
150 IFA$="I"THENB=1
160 IFA$="X"THENB=10
170 IFA$="C"THENB=100
180 IFA$="M"THENB=1000
190 IFA$="V"THENB=5
200 IFA$="L"THENB=50
210 IFA$="D"THENB=500
220 B(J)=B
230 J=J+1
240 NEXTI
250 FORJ=L-1TO0STEP-1
260 IFB(J)<B(J+1)THENS=S-B(J):GOTO280
270 S=S+B(J)
280 NEXTJ
290 PRINT R$;"=";S
310 GOTO110
```

Römische Zahlen sind für dieses kleine Basic-Programm, erprobt auf PET, CBM, AIM-65 und PC-100, keine Geheimcodes

und wer das Entstehungsjahr eines alten Prachtbaus wissen möchte, gerät leicht in Verlegenheit: Was bedeuten „L", „D" oder „C"?

Das römische Zahlensystem mit den Dezimalstufen I = 1, X = 10, C = 100, M = 1000, den Halbierungen V = 5, L = 50, D = 500 und der Aufforderung, bei Bedarf zu addieren, mutet recht modern an, jedenfalls gemessen an britischen Gewohnheiten; zählt man nun beflissen drauflos, so ist etwa LXXXX verboten, es muß eben XC heißen. Diese Vorschrift dient der Kürze, dennoch sind große Zahlen sehr lang und ohne Abakus schlecht deutbar.

Statt des antiken Rechenbretts kann man aber auch einen Mikrocomputer und das BASIC-Programm im *Bild* verwenden.

Wolfgang E. Mueller

Claus-Dieter Wittek

TRS-80-Telegrafie-Übungsprogramm

Ein großer Nachteil bei käuflichen Telegrafie-Lehrgängen sind die relativ kurzen Übungstexte. Bald weiß man die Texte auswendig, wodurch die Übungen schnell an Wirksamkeit verlieren. Das hier beschriebene Telegrafie-Programm umgeht diesen Nachteil.

Entweder kann ein vorgegebener Text ausgegeben werden oder aus einer Gruppe von Buchstaben, Ziffern und Satzzeichen werden mit Hilfe eines Zufallsgenerators 5er-Gruppen erzeugt. Hierbei können beliebige Zeichen vorgegeben werden.

Die Telegrafiezeichen werden in Geschwindigkeiten zwischen 26 BpM und 1000 BpM ausgegeben. Zusätzlich lassen sich zwischen die einzelnen Zeichen Pausen einschieben. Besonders bei Anfängern ist das von Vorteil, da dann die Zeichen z. B. mit einer Geschwindigkeit von 60 BpM gegeben werden, aber durch die Pausen die Gesamtgeschwindigkeit nur 40 BpM beträgt.

Das Programm (*Bild 1*) wurde für einen TRS-80 (Level II/4K) geschrieben. Vor Einladen des Programms ist MEMORY SIZE auf 19 500 zu setzen, damit das Maschinenprogramm nicht durch den Basic-Teil überschrieben werden kann.

Ist das Programm geladen und man schreibt „RUN", so dauert es einen Moment, bis sich der Rechner mit „TEXT ODER 5-ER GRUPPEN?" meldet. In dieser Zeit wird das Maschinenprogramm geladen.

Nach der Eingabe „TEXT" erwartet der Rechner einen bis zu 700 Zeichen umfassenden Text. Dabei dürfen nur folgende Zeichen verwendet werden: ABCDEFGHIJKLMNOPQRSTUVWXYZ 1234567890 ()=:-.,/?+ Die Texteingabe wird mit Drücken der ENTER-Taste abgeschlossen.

Bei einer anderen Eingabe als „TEXT" springt der Rechner in den 5er-Gruppen-Modus (z. B. wenn nur die ENTER-Taste gedrückt wird). Nun können Ziffern, Zeichen und Buchstaben eingegeben werden. Aus diesen Zeichen werden dann die 5er-Gruppen gebildet. Es sind maximal 100 Zeichen zulässig, d. h. Zeichen, die öfter erscheinen sollen, um diese besser üben zu können, werden häufiger eingegeben (z. B. dreimal das Alphabet und nur einmal die Zeichen und Ziffern).

Dann kann die gewünschte Anzahl von 5er-Gruppen eingegeben werden. Je nach Umfang der Gruppen muß man

```
2 REM CW-PROGRAMM   CLAUS-DIETER WITTEK, DF8QJ  30.8.79
3 REM MEMORY SIZE:  19500      FUER TRS-80 LEVEL II  4K / 16K
4 CLS:PRINT@64,'TELEGRAPHIE-PROGRAMM'
5 DATA50,82,1,53,76,94,85,54,32,33,35,39,47,63,62,60,56,48,120,1,1,46,1,115,1,5,30,26,14,3,27,
12,31,7,17,10,29,4,6,8,25,20,13,15,2,11,23,9,22,18,28,1,1,1,1,1
20 DATA0,197,213,229,221,229,38,79,17,128,60,221,33,66,76,221,126,0,167,202,90,79,198,116,111,
78,205,96,79,62,1,185,202,64,79,62,4,211,255,203,65,32,6,205,96,79,205,96,79,205,96,79,62,0,21
1,255,203,57,195,26,79,0,0,0,221,126,0,18,205,96,79
30 DATA205,101,79,205,101,79,205,101,79,19,221,35,195,15,79,0,0,0,0,221,225,225,209,193,201,58
,113,79,24,3,58,114,79,60,61,200,6,128,16,254,24,248,32,5,0,1
90 FORL=20380TO20403:READA:POKEL,A:NEXT:FORL=20404TO20435:READA:POKEL,A:POKEL+32,A:NEXT
100 FORL=20224TO20340:READA:POKEL,A:NEXT:POKE16527,79:POKE16526,1:FORL=20341TO20379:POKEL,1:NE
XT:FORL=20468TO20479:POKEL,1:NEXT:POKE20402,42:CLEAR200
210 CLS:PRINT:PRINT'TELEGRAPHIE-PROGRAMM':PRINT:FOR K=1TO3:POKE19521+K,86:NEXT:POKE19525,32:PO
KE19526,32
220 INPUT'TEXT ODER 5-ER GRUPPEN ';I$:O=1:IFI$='TEXT'THENO=2:GOTO2000
1010 PRINT'WELCHE ZEICHEN VERWENDEN (MAX.100) ':GOSUB3000:PRINT
1020 INPUT'WIEVIELE 5-ER GRUPPEN  (MAX.100) ';GR
1030 A=19527:FORL=1TOGR:FORK=1TO5:POKEA,ASC(MID$(J$,RND(LEN(J$)),1)):A=A+1:NEXT:POKEA,32:A=A+1
:NEXT
1040 PRINT:PRINT'MOEGLICHST 40-500 BPM':INPUT'WELCHE ZEICHEN-GESCHW. (BPM)';GO
1050 INPUT'WELCHE GESAMT-GESCHW. (BPM)';G
1070 POKE20337,6557/GO
1080 GX=17000/GO*(GO/G-1):IFGX)255PRINT'ZU EXTREME VARIATION':GOTO1040
1090 IFGX(0PRINT'ZEICHEN-GESCHW. ZU GERING':GOTO1040
1095 POKE20338,GX
1100 CLS:POKE A,32:POKEA+1,43:POKE A+2,0:U=USR(0)
1110 PRINT@960,;:INPUT'NEUE ZEICHEN (J/N)';I$:IFI$='J'THEN210
1115 IFI$=''THEN1020
1120 IFO=1INPUT'NEUE GRUPPEN (J/N)';I$:IFI$='J'THEN1020
1130 INPUT'NEUE GESCHWINDKEIT (J/N)';I$:IFI$='J'THEN1040ELSE1100
2000 A=19527:PRINT'TEXT-EINGABE:'
2010 I$=INKEY$:IFI$=''THEN2010ELSEI=ASC(I$)
2020 IFI=8THENPRINTI$;:A=A-1:GOTO2010
2030 IFI=13POKEA,0:PRINT:POKEA+1,0:GOTO1040
2035 IFA)20222PRINT:PRINT'SPEICHER-UEBERLAUF':GOTO2000
2040 POKEA,I:A=A+1:PRINTI$;
2050 GOTO2010
2999 END
3000 J$=''
3010 I$=INKEY$:IFI$=''THEN3010ELSEI=ASC(I$)
3020 IF(I=8)AND(LEN(J$))0)THENJ$=LEFT$(J$,LEN(J$)-1):PRINTI$;:GOTO3010
3030 IFI=13RETURN
3040 J$=J$+I$:PRINTI$;:GOTO3010
5000 REM
5010 REM
5020 REM     SONDERHEITEN DES AUSDRUCKS
5030 REM
5040 REM     EINZELNE ASCII-ZEICHEN MUSSTEN WIE FOLGT DARGESTELLT WERDEN:
5050 REM          AT-SYMBOL            ALS   KLEINES VIERECK
5060 REM          ANFUEHRUNGSSTRICHE   ALS   HOCHKOMMA
5070 REM          AUSRUFEZEICHEN       ALS   VIERECK AUF SPITZE
5080 REM
5090 REM     DIE ZEILEN 5000-5090 GEHOEREN NICHT ZUM PROGRAMM
```

Bild 1. Auflistung des Telegrafie-Übungsprogramms für den TRS-80, Level II/4 KByte

sich nun eine Weile gedulden, bis die Gruppen bestimmt sind.

Um bei relativ geringen Geschwindigkeiten die Klangbilder für höhere Tempi einüben zu können, wird im Programm zwischen Zeichen- und Gesamtgeschwindigkeit unterschieden. Sind beide gleich groß, so werden normgerechte Zeichen ausgegeben.

Ist die Gesamtgeschwindigkeit geringer als die Zeichengeschwindigkeit (z. B. Zeichengeschwindigkeit 80 BpM, Gesamtgeschwindigkeit 40 BpM), so werden die Zeichen in Tempo 80 BpM gegeben. Durch Einfügen von Pausen

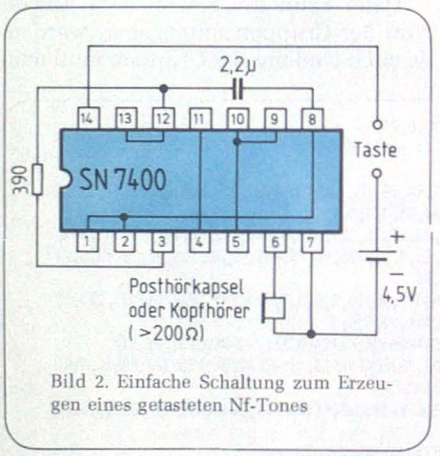


Bild 2. Einfache Schaltung zum Erzeugen eines getasteten Nf-Tones

werden aber nur insgesamt 40 Zeichen je Minute ausgegeben. Dies ist besonders für den Anfänger empfehlenswert, da bei geringer Schreibbelastung schon die schnellen Klangbilder eingeübt werden.

Um eine Kontrolle über seine Fehler zu haben, wird, nachdem das Zeichen ausgegeben wurde, dieses auf den Bildschirm geschrieben. Zusätzlich können der gleiche Text bzw. die gleichen 5er-Gruppen erneut mit gleicher oder geänderter Geschwindigkeit ausgegeben werden.

Der Rechner kann wie jede Telegraphie-Taste angeschlossen werden. In der hier beschriebenen Form wird das Reed-Relais der Kassettenmotor-Steuerung zur Ausgabe der Zeichen benutzt. Dabei ist auch gleich eine galvanische Entkopplung vom Rechner gegeben. Allerdings sollte man beachten, daß das Reed-Relais nur kleine Ströme steuern kann.

Für Übungszwecke kann mit wenigen Bauteilen ein simpler Tongenerator aufgebaut werden (*Bild 2*). Die Schaltung bedarf wohl keiner weiteren Erläuterung.

Nascom-Texteditor

Der in FUNKSCHAU 1980, Heft 1, Seite 87 beschriebene Basic-Texteditor läuft nicht ohne Änderungen auf dem 8-KByte-Microsoft-Basic des Nascom-1. Um dieses recht nützliche Programm auch auf diesem Microcomputer laufen zu lassen, wurde das hier beschriebene Programm (*Bild*) geschrieben.

Zusätzlich wurde noch eine Ergänzung eingefügt, die es ermöglicht, daß beim Nascom-1 eine permanente Überschrift in der Kopfzeile erzeugt wird. In diese Überschrift wird jeweils der aktuelle Modus des Editors vom Programm mit eingetragen.
Da das Nascom-Microsoft-Basic keinen GET$-Befehl kennt, mußte dieser durch eine INPUT-Anweisung ersetzt werden. Als Sicherheit gegen Fehlbedienung wurden noch IF...Then-Bedingungen hinzugefügt, da als Eingabe nur ein Buchstabe erwartet wird.

Zur Datenspeicherung wird hier der normale, externe Kassettenrecorder herangezogen, so wie er auch zur Programmabspeicherung benutzt wird. Welche Betriebsart (Aufnahme oder Wiedergabe) benötigt wird, fordert das Programm durch entsprechende Print-Befehle an. Als Ergänzung wird bei jeder verlangten Kassetten-Aktivität, die Drive-LED angeschaltet, die dieses signalisiert. Der Recorder ist wieder abzuschalten, sowie die LED erloschen ist.

Gegebenfalls kann der Recorder ferngesteuert werden, indem ein Relais in Reihe mit der LED geschaltet wird, dessen Kontakte dann den Recorder an- und ausschalten. Als System-Monitor wird die Ausführung NASBUG T4 benötigt.

Winfried Widmann

```
 1 CLS:CLEAR8000:REM UEBERSCHRIFT IN KOPFZEILE
 2 PRINT" * * * * * * E D I T O R - MODE   * * * * * * ";
 3 FORC=2954TO3000STEP2
 4 DOKE+64,DEEK(C)
 5 NEXTC
 6 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
 7 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
 8 REM EDITOR-FUNKSCHAU HEFT1/80 VERSION 2.1
 9 REM MODIFIED BY WINFRIED WIDMANN FOR NASCOM-1
10 REM VERSION2.2-14.01.80
40 INPUT"SECTION NRS.? Y/N;S$:IFS$=""THEN40
50 IFLEN(S$)>1THEN40
120 RESTORE:INPUT"MODE";M$:IFM$=""THEN120
125 IFLEN(S$)>1THEN120
126 SCREEN33,16:PRINTM$:REM MODE WRITE IN TOP
180 S=0:I=-1
450 I=J:INPUT"KILL LINE? Y/N";M$:IFM$=""THEN450
500 INPUT"CLEAR LAST FILE? Y/N";M$:IFM$=""THEN500
510 IFLEN(M$)>1THEN500
540 PRINT"TURN CASSETTE-TAPE ON, MODE START":GOSUB980
545 DOKE4100,1146:DOKE3084,0:U=INP(USR(1)):INPUTX$
550 INPUTA$(J):IFA$(J)<>"END"THENJ=J+1:GOTO550
560 DOKE4100,1984:U=INP(USR(1)):PRINT"LOADED":GOSUB980
640 IFLEN(M$)=1THENM$="A"+M$
700 J=0:PRINT"TURN CASSETTE-TAPE ON, MODE REC!!!!"
705 GOSUB980
710 DOKE4100,1146:DOKE3084,0:U=INP(USR(1))
715 INPUT"IF TAPE ON, THEN PUSH KEY NEW LINE!";Z$:IFZ$<>""THEN715
718 PRINT:Z=1
719 Z=Z+1:IFZ<>30THEN719
720 PRINTA$(J):Z=1
721 Z=Z+1:IFZ<>30THEN721
722 IFA$(J)<>"END"TEHNJ=J+1:GOTO720
725 J=0:DOKE4100,1984:U=INP(USR(1)):GOSUB980
726 RETURN
760 INPUT"CUT OFF NR. FIELD? Y/N";M$:IFM$=""THEN760
770 IFLEN(M$)>1THEN760
980 DOKE4100,3152:DOKE3152,4158:DOKE3154,20941:DOKE3156,-14080
990 U=INP(USR(1)):RETURN
```

Diese Zeilen müssen geändert werden, will man den Basic-Texteditor auf Heft 1/1980 auf dem Basic-Interpreter des Mikrocomputers Nascom-1 laufen lassen

Basic-Texteditor

Bereits in Heft 1/1980 veröffentlichten wir einen in Basic geschriebenen Texteditor, der zu den universellsten Programmen überhaupt gehört. Im Laufe der Zeit ergaben sich einige Verbesserungen und Erweiterungen, die auch eine komfortable Dialog-Bedienung einschließen. Deshalb drucken wir hier die neue Editor-Version 4.1 für die CBM-Doppelfloppy 3040 ab. Die Kombination mit der auf der Seite 93 beschriebenen INSTRING-Funktion folgt im nächsten Heft.

Wie der in Heft 1/1980 beschriebene Editor dient auch diese Version vorwiegend zum Erstellen von Verzeichnissen aller Art: Zeitschriften-Inhaltsverzeichnisse, Lagerlisten usw. Besonders praktisch ist dabei die Möglichkeit, nach Stichworten zu suchen, die irgendwo innerhalb einer Textzeile stehen können. So wird etwa die Zeile „MAX MEIER 123654", die Teil eines Telefonnummern-Verzeichnisses ist, ausgegeben, wenn man als Stichwort „MAX", „MEIER", „MEI", „123654" oder auch nur „36" eintippt.

Ferner ist die Unterteilung einer Textfile (das ist die Gesamtheit der gespeicherten Textzeilen) in Rubriken möglich. Hierzu wird eine aus max. zwei Zeichen bestehende Rubrikkennzeichnung verwendet. Beim alphabetischen Sortieren wird dann automatisch nach Rubriken getrennt sortiert.

Eine Besonderheit ist auch, daß man beim Programmstart die gewünschte Druckbreite angeben kann. Der Befehl P

```
10 PRINT"s********** e d i t o r **********":PRINT:REM VERSION4.1
20 INPUT"dRUCKER-zEILENBREITE";W:D=600:REM MAX.ZEILENZAHL
40 PRINT"rUBRIKFELDER? J/N ";
50 GETS$:IFS$=""GOTO50
60 PRINT"R"S$"r":DIMA$(D):DATAA,B,D,F,I,K,L,N,P,R,S,T,U:GOSUB950
110 PRINT"RbEFEHL?  ";
120 RESTORE:GETM$:IFM$=""GOTO120
130 FORI=1TO13:READB$:IFB$=M$THENPRINT"I"B$"r":GOTO160
140 NEXT:GOTO120
160 ONIGOSUB180,230,250,280,380,450,500,580,660,760,780,730,740
170 PRINTA$(J):GOTO110
180 S=0:I=-1:GETM$:IFM$=" "THENPRINT"sORTIERVORGANG UNTERBROCHEN":RETURN
190 I=I+1:IFA$(I+1)="END"ANDS=0THENPRINT"sORTIERT":RETURN
200 IFA$(I+1)="END"GOTO180
210 IFA$(I))A$(I+1)THENM$=A$(I):A$(I)=A$(I+1):A$(I+1)=M$:S=1
220 GOTO190
230 FORJ=0TOD:IFA$(J)="END"THENRETURN
240 NEXT
250 IFA$(J)<>"END"THENJ=J+1
260 RETURN
280 J=-1
290 INPUT"sUCHBEGRIFF=";B$:K=LEN(B$)
300 J=J+1:IFA$(J)="END"THENPRINT:RETURN
310 S=LEN(A$(J))-K:IFS<0GOTO300
320 FORI=1TOS+1:M$=MID$(A$(J),I,K):IFM$=B$GOTO360
340 NEXT:GOTO300
360 PRINTA$(J)
362 GETE$:IFE$<>" "GOTO300
364 PRINT"RF=fORTSETZUNG,S=sTOPr ";
366 GETE$:IFE$="S"THENPRINT"stop":RETURN
368 IFE$="F"THENPRINT" ok":GOTO300
369 GOTO366
380 I=-1:REM INSERT
390 I=I+1:IFA$(I)<>"END"GOTO390
410 A$(I+1)=A$(I):I=I-1:IFI>=JGOTO410
420 INPUTM$:IFM$="END"THENI=J:GOTO480
425 IFLEFT$(M$,1)=" "ORLEFT$(M$,1)="?"THENM$=MID$(M$,2):GOTO425
430 IFS$="J"THENM$="** "+M$
440 A$(J)=M$:J=J+1:GOTO380
450 PRINT"lOESCHEN? J/N":I=J
460 GETM$:IFM$<>"J"GOTO460
470 IFM$<>"J"THENRETURN
480 I=I+1:IFA$(I-1)="END"THENRETURN
490 A$(I-1)=A$(I):GOTO480
500 IFA$(0)="END"GOTO540
505 PRINT"bISHERIGE tEXTE LOESCHEN? J/N ";
510 GETM$:IFM$=""GOTO510
520 PRINT"R"M$:IFM$="J"THENGOSUB950
530 GOSUB230
540 INPUT"fILENAME=";M$:PRINT"dISK (1/0)? ";
541 GETE$:IFE$=""GOTO541
542 PRINT"R"E$;:IFE$<"0"ORE$>"1"THENRETURN
545 M$=E$+":"+M$:OPEN15,8,15:OPEN1,8,0,M$:PRINT"r lADEVORGANG ";
550 INPUT#1,A$(J):IFASC(A$(J))<33THENA$(J)=MID$(A$(J),2)
560 IFA$(J)<>"END"THENJ=J+1:GOTO550
570 GOTO720
580 IFS$="J"GOTO620
600 S$="J":FORI=0TOD:IFA$(I)="END"GOTO620
610 A$(I)="** "+A$(I):NEXT
620 IFA$(J)="END"THENRETURN:REM DEF NR
630 INPUT"rUBRIK=";M$:A$(J)=MID$(A$(J),3)
640 IFLEN(M$)=1THENM$="#"+M$
650 A$(J)=M$+A$(J):S$="J":GOTO250
660 PRINT"aUSDRUCK AB DER AKTUELLEN zEILE":PRINT"dRUCKER? J/N":R$=""
663 GETE$:IFE$=""GOTO663
664 IFE$="J"THENOPEN1,7:PRINT#1,CHR$(2):GOTO670
665 IFE$<>"N"THENRETURN
670 FORI=JTOD:IFA$(I)="END"THENPRINT:CLOSE1:RETURN
680 IFA$(I))""THENGOSUB820
685 GETF$:IFF$<>" "THENNEXTI
686 PRINT"RF=fORTSETZUNG,S=sTOP
687 GETF$:IFF$="F"THENNEXTI
688 IFF$=""GOTO687
689 CLOSE1:RETURN
700 INPUT"fILENAME=";M$:PRINT"dISK (1/0)? R";
701 GETE$:IFE$=""GOTO701
702 IFE$<"0"ORE$>"1"THENRETURN
705 M$=E$+":"+M$:OPEN1,8,1,M$:J=0:OPEN15,8,15:PRINTE$"r ";
710 PRINT#1,A$(J):IFA$(J)<>"END"THENJ=J+1:GOTO710
720 INPUT#15,M$,B$,C$,D$:PRINTB$:CLOSE1:CLOSE15
730 J=0:RETURN
740 IFJ>0THENJ=J-1
750 RETURN
760 PRINT"rUBRIKFELD LOESCHEN? J/N"
770 GETM$:IFM$=""GOTO770
780 IFM$<>"J"THENRETURN
790 S$="N":PRINT"gELOESCHT"
800 FORI=0TOD:IFA$(I)="END"THENRETURN
810 A$(I)=MID$(A$(I),4):NEXT
820 M$=A$(I):B$=LEFT$(M$,2):IFS$="N"GOTO850
830 IFB$<>R$THENR$=B$:PRINT:PRINTR$:IFE$="J"THENPRINT#1:PRINT#1,R$
840 M$=MID$(M$,4)
850 IFLEFT$(M$,1)=" "THENM$=MID$(M$,2):GOTO850
860 IFLEN(M$)THENC$=M$:GOTO970
870 K=W+1
880 K=K-1:IFK=0THENK=W:GOTO920
890 B$=MID$(M$,K,1):IFB$>"?"GOTO880
900 IFB$<>":"ANDB$>","GOTO880
910 IFB$="'"ORB$="("GOTO880
920 C$=LEFT$(M$,K):GOSUB970
930 M$=MID$(M$,K+1):PRINT" ";:IFE$="J"THENPRINT#1," ";
940 GOTO850
950 FORI=0TOD:A$(I)="":NEXT
960 J=0:A$(J)="END":PRINT"sPEICHER GELOESCHT":PRINT:RETURN
970 D$="":PRINTC$:IFE$<>"J"THENRETURN
975 FORS=1TOLEN(C$):B$=MID$(C$,S,1)
980 IFASC(B$))128THENB$=CHR$(ASC(B$)-128):GOTO990:REM NICHT BEI CBM-DRUCKER
985 IFASC(B$))64THENB$=CHR$(ASC(B$)+128):REM NICHT BEI CBM-DRUCKER
990 D$=D$+B$:NEXT:PRINT#1,D$:RETURN
```

Listing des Texteditors in der CBM-Syntax. Die unterstrichenen Zeichen sind Steuerzeichen: s = Bildschirm löschen, R = Reverse, r = Reverse off, I = Cursor nach links. Die maximale Zeilenzahl (Programmzeile 20, hier 600) sollte man bei der 8-K-Version des Computers geringer wählen, da sonst ein Speicherüberlauf auftritt. Zu beachten ist, daß der für das Listing verwendete Drucker Klein- und Großbuchstaben vertauscht – daher auch die Zeilen 980 und 985

(vgl. *Tabelle*) sorgt dann nicht nur für die Einhaltung dieser Zeilenlänge beim Ausdrucken, sondern kümmert sich auch um die richtige Trennung bei Leerräumen, Satzzeichen und Bindestrichen, so daß keine Worte „zerrissen" werden.

Das im *Bild* vollständig aufgelistete Programm ist für einen Computer CBM 3032 nebst Doppel-Floppy-Station CBM 3040 geschrieben. Der Autor verwendete einen Centronics-Drucker 779, der sich stur an die ASCII-Zeichennorm hält und deshalb Groß- und Kleinbuchstaben beim Ausdrucken vertauscht (der CBM 3032 weicht in dieser Beziehung von der ASCII-Norm ab). Deshalb enthält das Programm in der Druckroutine zwei Zeilen (980 und 985), die das wieder in Ordnung bringen. Geeignete Routinen für die Verwendung von Kassetten-Laufwerken finden sich, ebenso wie eine ausführliche Beschreibung des Ur-Editors, in Heft 1/1980 und 3/1980.

Herwig Feichtinger

Befehlsvorrat des Editors

A Alphabetisches Sortieren aller Textzeilen, ggf. auch nach Rubriken
B Letzte Textzeile anzeigen („bottom"; immer END)
D Eine Zeile nach unten gehen („down")
F Stichwortsuche in allen Textzeilen („find")
I Neue Zeile(n) vor die zuletzt angezeigte eingeben („insert"); mit „end" abschließen
K Angezeigte Zeile löschen; die folgenden rücken nach („kill")
L File von der Floppy laden („load")
N Angezeigte Zeile mit Rubrikbezeichnung versehen (zwei Zeichen)
P Alle Zeilen ab der zuletzt angezeigten formatiert und ggf. mit Rubrik-Überschriften ausgeben, ggf. auch auf Drucker („print")
R Die Rubrikbezeichnung, d. h. die ersten drei Zeichen jeder Zeile löschen („remove")
S Die gesamte Textfile auf Floppy abspeichern („save")
T Erste (oberste) Textzeile anzeigen („top")
U Eine Zeile nach oben gehen („up")

Bei den Befehlen A, F und P ist ein Unterbrechen der Ausführung jederzeit durch Antippen der Leertaste möglich; nach einer kurzen Wartezeit erscheint dann im Bildschirm die Anweisung, wie man fortfahren kann.
Ein „Aussteigen" aus den Routinen L und S ist möglich, indem man für die Disk-Nummern nicht 0 oder 1, sondern eine ungültige Bezeichnung eingibt.
Die Texteingabe erfolgt grundsätzlich mit dem I-Befehl, auch dann, wenn noch kein Text im Speicher steht. Die Textzeilen dürfen keine Kommas oder Doppelpunkte enthalten.

Einfach-Renumber beim AIM-65-Basic

Das kleine Maschinenprogramm im *Bild* numeriert die Zeilen eines AIM-Basic-Programms in Zehnerschritten neu. Wegen der Kürze des Programms erfolgt dabei allerdings keine Sprungadressen-Korrektur – dies wird dem Benutzer überlassen. Die Schrittweite kann an der Adresse 0FC4 verändert werden.

Die Verwendung geschieht wie folgt: Basic ist zunächst mit MEMORY SIZE = 3989 zu initialisieren. Dann geht man mit der Escape-Taste in den Monitor zurück und lädt das Programm laut Listing. Nun kann man mit Taste „6" in Basic zurückgehen, das neu zu numerierende Basic-Programm laden und durch Verwendung von NUM als Direktbefehl umnumerieren.

Der „Trick" ist hier, daß das Renumber-Programm in die Zero-Page-Routine eingreift, mit der der Basic-Interpreter Zeichen holt und so das Befehlswort NUM erkennen kann.

E. Milinkowitsch

```
<K>*=00C1
/01
 00C1 D0 BNE 010F

<K>*=010F
/01
 010F 4C JMP 0F95

<K>*=0F95
/44
 0F95 A5 LDA 16
 0F97 C9 CMP #4E
 0F99 F0 BEQ 0F9E
 0F9B 4C JMP 00C5
 0F9E A5 LDA 17
 0FA0 C9 CMP #55
 0FA2 F0 BEQ 0FA7
 0FA4 4C JMP 00C5
```

```
0FA7 A5 LDA 18
0FA9 C9 CMP #4D
0FAB F0 BEQ 0FB0
0FAD 4C JMP 00C5
0FB0 20 JSR EB4D
0FB3 A5 LDA 73
0FB5 8D STA 0FF8
0FB8 A5 LDA 74
0FBA 8D STA 0FF9
0FBD A2 LDX #01
0FBF 20 JSR 0FE0
0FC2 90 BCC 0FFD
0FC4 A9 LDA #0A
0FC6 20 JSR E54E
0FC9 A9 LDA #9D
0FCB 20 JSR 0FE8
0FCE A2 LDX #00
0FD0 20 JSR 0FE0
```

```
0FD3 8D STA 0FF8
0FD6 E8 INX
0FD7 20 JSR 0FF7
0FDA 8D STA 0FF9
0FDD 4C JMP 0FC2
0FE0 A9 LDA #BD
0FE2 8D STA 0FF7
0FE5 4C JMP 0FF7
0FE8 8D STA 0FF7
0FEB A2 LDX #02
0FED AD LDA A41E
0FF0 20 JSR 0FF7
0FF3 E8 INX
0FF4 AD LDA A41F
0FF7 BD LDA 0044,X
0FFA C9 CMP #02
0FFC 60 RTS
0FFD 4C JMP B003
```

Eine Neunumerierung von Basic-Programmzeilen in Zehnerschritten erlaubt dieses kurze Maschinenprogramm. Es wird durch Eingabe von NUM als Basic-Direktbefehl angesprochen und zeigt, wie einfach die Erweiterung des Basic-Sprachschatzes ist, indem man die Zeichenhol-Routine des Interpreters bei 00C1 „knackt"

µC-Nachrichten

Sowohl für den Apple-II als auch für den AIM-65 ist jetzt die Programmiersprache **FORTH** erhältlich. Diese Sprache ist recht maschinennah; ein Hauptvorteil ist, daß der Anwender selbst den Befehlsvorrat erweitern kann.

□

Der **Nachfolger des AIM-65**, ein Computer mit 32-KByte-RAM und einem 40stelligen Fluoreszenz-Display wird voraussichtlich ab Anfang nächsten Jahres von Rockwell erhältlich sein. Ob auch Siemens (wie bei AIM-65/PC-100) den Vertrieb übernehmen wird, steht noch nicht fest.

□

Als Zusammenfassung der interessantesten Beiträge aus den Ausgaben 1-6 der 6502-Zeitschrift **„65XX Micro Mag"** brachte Roland Löhr das „Buch 1-6 des 65XX Micro Mag" heraus. Es kann direkt vom Herausgeber (Hansdorfer Straße 4, 2070 Ahrensburg) bezogen werden.

Reinhold Martin

Schnelle Stichwortsuche beim CBM

Haben Sie schon einmal versucht, einen String in einem längeren String zu suchen? Dies ist in Basic nicht nur relativ mühsam zu programmieren, sondern aufgrund der hierzu nötigen FOR-NEXT-Schleife auch sehr zeitaufwendig. Im folgenden wird eine INSTRING-Funktion vorgestellt, die sehr schnell arbeitet.

Die Funktion INSTRING weist der Ergebnis-Variablen die Position des ersten Zeichens des Suchstrings im Quellstring zu. Wird der Suchstring nicht gefunden, ist das Ergebnis Null. Den beiden Stringvariablen muß vorher ein String zugewiesen worden sein, sonst meldet der Rechner einen STRING NOT FOUND ERROR. Als Illustration können die dargestellten Beispiele dienen. Das vor dem Befehl stehende Sonderzeichen „@" darf auf keinen Fall weggelassen werden, da sonst ein SYNTAX ERROR auftritt, weil die neue Funktion für den Basic-Interpreter vollkommen unbekannt ist. Durch das vorausgehende Sonderzeichen und die Änderung einer im RAM befindlichen Interpreter-Routine war die Einführung des Befehls jedoch möglich. Die Syntax der Funktion sieht folgendermaßen aus:

10 A=@INST(Q$,S$,P)

bzw.

10 A=@INST(Q$,S$)

A = Zielvariable
Q$= Quellstring
S$= Suchstring
P = Position, ab der gesucht werden soll.

Die Wahl der Variablen ist belanglos. Es können alle im Basic erlaubten Variablennamen benutzt werden, auch Arrays und Integer-Variablen.

Eine Einschränkung ist jedoch zu beachten: Die Funktion darf keiner IF-Anweisung folgen (wohl aber dem THEN). Diese kleine Einschränkung kann umgangen werden, indem zunächst die Instring-Funktion ausgewertet und erst danach der logische Vergleich (mit IF) durchgeführt wird. Beispiel:

```
10 A =@ INST(Q$, S$)
20 IF A=...THEN...
```

Während sich die Bedeutung der einzelnen Variablen aus dem Listing ergibt, bedarf die den beiden Stringvariablen folgende Variable P (optional) einer kurzen Erläuterung. Diese Variable gibt an, ab welcher Position der Suchstring im Quellstring gesucht werden soll. Dies ist dann notwendig, wenn der Quellstring den Suchstring mehrfach enthält, da die Funktion sonst nur die Position des ersten Auftretens des Suchstrings ergibt. Wenn die Positionsvariable weggelassen wird, wird ab Beginn des Quellstrings gesucht.

Nach dem Starten des Programms (*Bild 2*) ist die Funktion aktiv und steht zur Programmierung zur Verfügung. Es sollte als erstes Programm geladen werden und da sie (je nach Speichergröße) am Ende des RAM-Bereichs liegt, sollte vorher ein Reset erfolgen, z. B. Ausschalten des Rechners.

```
40 DIM Q$(10),B$(5)
45 Q$(5)="DIES IST EIN EINFACHES PROG"
50 S$(5)="EIN"
55 A=@INST(Q$(5),S$(5))
60 PRINT A
65 P%=12
70 A=@INST(Q$(5),S$(5),P%)
75 PRINT A
```

Bild 1. Programmierbeispiel mit der neuen INST-Funktion. Sie findet einen Suchbegriff (hier S$) in einem zu untersuchenden Quellstring (hier die Array-Variable Q$) und druckt seine Position aus. Mit der Integer-Variablen P% kann man die Suche bei einer bestimmten Zeichenposition von Q$ beginnen lassen

Bild 2. ▶
Initialisierungsprogramm für die Funktion INST. Es generiert nach dem Start ein am RAM-Ende liegendes Maschinenprogramm. Wichtiger Hinweis: Dieses Programm ist nur auf dem CBM-Betriebssystem lauffähig, nicht auf dem PET-2001-System!

```
450 PRINT"sBITTE WARTEN"
460 X=7821:Y=PEEK(53)-32:X=256*Y+X
470 FOR I=0 TO 370
480 READ A:IFA<0THENA=Y-A
490 POKE (X+I),A
500 NEXT I
510 SYS(7821+256*Y)
520 PRINT"sINSTRING IST INITIALISIERT"
530 NEW
540 REM
550 REM DATA PROG
560 REM
570 DATA 169,141,133,052,169,-30,133,053,169,076,133,121,169,162,133,122,169
580 DATA -30,133,123,096,201,064,240,036,201,058,176,247,076,125,000,230,119
590 DATA 208,002,230,120,096,165,119,208,002,198,120,198,119,096,032,173,-30
600 DATA 208,003,032,180,-30,160,000,177,119,096,164,120,192,002,240,214,169
610 DATA 000,133,177,133,078,032,194,-30,201,178,240,003,076,003,206,032,142
620 DATA 204,165,008,072,165,068,133,087,165,069,133,088,032,173,-30,160,005
630 DATA 032,112,000,217,242,-31,208,226,136,208,245,032,189,-30,032,189,207
640 DATA 032,144,204,032,223,-31,160,000,177,068,133,080,200,177,068,141,122
650 DATA -31,141,157,-31,200,177,068,141,123,-31,141,158,-31,032,248,205,032
660 DATA 189,207,032,144,204,032,223,-31,160,000,177,068,133,083,200,177,068
670 DATA 141,119,-31,141,152,-31,200,177,068,141,120,-31,141,153,-31,032,197
680 DATA -30,201,041,240,039,201,044,240,003,076,003,206,032,173,-30,032,159
690 DATA 204,032,142,204,032,154,208,165,098,133,177,197,080,144,002,176,060
700 DATA 032,197,-30,201,041,240,003,076,003,206,166,177,173,048,004,221,034
710 DATA 004,240,009,232,228,080,208,243,162,000,240,029,134,079,169,000,133
720 DATA 089,230,079,230,089,164,089,196,083,240,012,185,048,004,164,079,217
730 DATA 034,004,208,221,240,234,232,134,078,165,087,133,068,165,088,133,069
740 DATA 104,016,014,160,000,169,000,145,068,200,165,078,145,068,076,212,-31
750 DATA 169,000,133,095,164,078,132,096,162,144,056,032,085,219,166,068,164
760 DATA 069,032,227,218,104,104,104,104,032,173,-30,076,196,198,096,224,000
770 DATA 208,251,162,008,189,247,-31,032,210,255,202,208,247,160,040,076,125
780 DATA 245,040,084,083,078,073,071,078,073,082,084,083,063,013,141,133,052
```

Hans-Georg Joepgen

„HERE" sorgt für schnelle Sprünge

Vorschlag für ein neues Basic-Statement

Seit es Basic gibt, kommen aus dem Kreis der Benutzer immer neue Forderungen nach Erweiterungen dieser Programmiersprache. Ein Teil der Wünsche ist tatsächlich in die heute verbreiteten Basic-Dialekte eingeflossen und hat der Sprache ihre hohe Leistungsfähigkeit verliehen, die sie heute hat und die nur noch bedingte Ähnlichkeiten mit „Ur-Basic" zeigt, in dem es nicht einmal eine Möglichkeit zur unmittelbaren Dateneingabe gab. Was heute INPUT und GET holen, war in DATA-Zeilen zu deponieren – dies nur ein Beispiel von mehreren möglichen.

Heute hat Basic einen Reifegrad und ein Standardisierungsmaß erreicht, der es nicht als ratsam erscheinen läßt, leichtfertig neue Erweiterungen zu fordern: Es gibt schier schon zu viele Firmendialekte. Das tut nicht gut; überdies zeigt eine kritische Betrachtung der noch immer laut werdenden Änderungswünsche, daß sich der angestrebte Zweck, wenn bisweilen auch etwas unbequem, in den weitaus meisten Fällen durch eine Kombination bereits in der Sprache enthaltener Elemente verwirklichen läßt.

Wenn der Verfasser an dieser Stelle ein solches neues Statement vorschlägt, so tut er's (fast) ein wenig gegen seine grundsätzliche Überzeugung, daß an Standard-Basic möglichst nicht zu rühren sei. Die Rechtfertigung für den Vorschlag kommt aus zweierlei Tatsachen. Zum einen ist das geforderte Statement *nicht* aus Standardelementen der Sprache zu verwirklichen, zum zweiten könnte es zu einer ganz beträchtlichen Änderung der Sprachstruktur in Richtung höhere Transparenz, leichtere Erkennbarkeit und bessere Dokumentierungsmöglichkeit führen, besonders dann, wenn das neue Statement in einem jener modernen Compiler oder Interpreter zur Verfügung gestellt würde, die ungekürzte Variablennamen über viele Zeichen hinweg erlauben wie zum Beispiel die von Optimized Systems Software in Cupertino für die Maschinen der Apple-Klasse angebotenen Version.

Genug der Vorreden: Was soll HERE – diesen Namen schlägt der Verfasser vor – bewirken? Mit einem Satz: Die Zuordnung der aktuellen Zeilennummer zu einer Variablen, deren Namen dem Kommandowort HERE (englisch „hier") folgt. Konkret: Eine oft benutzte Subroutine, sie möge der Fehlermeldung nach Tastatureingaben dienen oder einem ähnlich gutem Zweck, habe die Zeilennummer 1234. Zusammen mit anderen Subroutinen, die ihre eigenen Zeilennummern haben, sitzt sie also im Programm. Soll sie benutzt werden, so erfolgt ihr Aufruf durch GOSUB 1234.

Da es nun eine ganze Anzahl solcher Subroutinen im Programm gibt und sie von vielerlei Zeilen aus gerufen werden, kommt man, größere Programme vorausgesetzt, beim Verfassen, Ändern oder auch nur beim Analysieren nicht darum herum, sich eine Art Spickzettel anzufertigen, ein Verzeichnis darüber, was eigentlich unter welcher Zeilennummer passiert. Mit HERE käme man um einen Teil dieser Mühe herum. Dann würde das erste Statement unserer Subroutine zur Fehlerbehandlung nämlich heißen: HERE FEHLERCHECK, gefolgt von den Arbeitsanweisungen. Und jene Programmsegmente, die der Hilfe der Fehlerbehandlungsroutine bedürfen, würden nicht länger lauten: GOSUB 1234 oder IF NENNER = 0 THEN 1234, hier hieße es schön verständlich: GOSUB FEHLERCHECK, THEN FEHLERCHECK und so fort.

Wenn die Namen der Subroutinen, Rücksprungpunkte oder Verzweigungsziele klug gewählt werden, dann ergibt sich eine gleichsam selbstdokumentierende Programmstruktur. Aber HERE bietet noch weitere Vorteile: Bei Programm-Umbauten mit Änderungen der Zeilennummern wird das Leben viel leichter. Wenn FEHLERCHECK von Zeile 1234 nach 2345 umzieht und es dann dort heißt, HERE FEHLERCHECK, dann muß an all jenen GOSUBs, GOTOs und IF...THENs mit Bezug auf FEHLERCHECK absolut nichts geändert werden. Das all dies der Verbesserung der Erlernbarkeit der Sprache dient und die Fehlerwahrscheinlichkeit beim Programmieren herabsetzt, wird wohl nicht zu bestreiten sein.

Soweit also die theoretische Vorstellung des Vorschlages und die Begründung seiner Zweckmäßigkeit. Steigen wir nun in die Niederungen des Interpreterbaues herab und denken wir kurz darüber nach, wie HERE denn technisch konkret verwirklicht werden könnte. Es muß, soviel ist klar, eine gesonderte Tabelle im Interpreter existieren, die beim Antreffen eines HERE-Statements eine Variable mit dem Namen errichtet, FEHLERCHECK in unserem Beispiel. Weiterhin muß sich der Interpreter aus dem Pointer für die aktuelle Zeilennummer den richtigen Zeilennummer-Wert abkupfern und ihn der Variablen FEHLERCHECK zuorden, rein in die Tabelle, Deckel zu. Wird's dann ernst, soll das Benutzerprogramm ablaufen, dann hat der Interpreter nach Sprunganweisungen nur noch in dieser Sprungzieltabelle nachzuschauen, welche Zeile denn nun zu FEHLERCHECK gehört: Und dort, bitteschön, weiterzuschaffen. All dies ist kein Hexenwerk – im Vertrauen, so oder so ähnlich gehen FORTRAN-Compiler mit ihren Labels und P-Code-Interpreter mit ihren Pascal-Funktionen und Prozeduren um. Also, welches Software-Haus oder welcher Mikrocomputer-Hersteller wagt's?

Es gibt schon viel zu viele Basic-Dialekte, hieß es eingangs. Und so kann denn auch jemand, der das Geschehen in der Szene sonst an sich gewissenhaft verfolgt, nicht jede der zahlreichen Spielarten kennen. Denkbar wäre somit, daß irgendjemand HERE – vielleicht unter anderem Namen – schon eine ganze Weile hat. Für diesen Fall bittet der Verfassen, diesen Artikel ein wenig umzudeuten: In die Frage nämlich, warum uns die Hersteller der heute in großen Stückzahlen verkauften Massen-Mikrocomputer die wunderschönen Möglichkeiten von HERE vorenthalten und sich nicht schleunigst eine Lizenz für diese Interpreter-Erweiterung besorgen.

Autostart bei KIM-Kassetten

Manchmal ist es durchaus sinnvoll, wenn direkt nach dem Laden eines Programmes von der Kassette dieses sich ohne weiteres Zutun selbst starten kann.

Im Prinzip ist dies auch beim KIM-1 möglich. Ein entsprechendes Ladeprogramm muß allerdings bereits im Arbeitsspeicher vorhanden sein. Die Startadresse am Anfang des Datenblocks wird damit festgehalten. Nach Beendigung des Ladens wird nicht in den Monitor gesprungen, sondern auf die anfangs in der Zero-Page festgehaltenen Adresse mit einem JMP-Indirekt-Befehl. Es ist aber nötig, dieses Programm vorher zu laden. Von einem reinen Autostart nach dem Einschalten des KIMs kann dann nicht die Rede sein.

Und doch ist ein Autostart möglich, wenn dafür gesorgt wird, daß das erwähnte Programm sich ebenfalls selbst startet. Einzige Vorbereitung nach dem Einschalten des KIMs: Die Adresse 17F9 muß mit 00 oder mit einer entsprechenden ID-Nummer versehen sein. An der Adresse 1873 wird gestartet und der Kassettenrecorder eingeschaltet. Nach Durchlauf zweier Datenblöcke startet sich das geladene Programm selbst und schaltet, wenn dies im Programm so vorgesehen ist, den Kassettenrecorder ab.

Die Lösung für diese Betriebsart liegt in einer entsprechenden Initialierung des „Volatile Execution-Blocks", kurz des VEB-Bereiches bei den Adressen 17EC-17EE, die für eine Veränderung des Programmablaufes in dem Augenblick sorgt, wenn die Adresse 17EC mit Daten von der Kassette beschrieben wird. Dies kann dadurch geschehen, daß das Autostartprogramm auf den Adreßbereich ab 1780 gelegt wird, wobei beim Laden dieses Programms das entscheidende Byte bei 17EC mit berücksichtigt wird.

```
0280  20 32 19 a2 06 bd df 17
0288  9d eb 17 ca d0 f7 4c 8c
0290  18 20 ea 19 a2 02 20 24
0298  1a c9 2f f0 0e 20 00 1a
02a0  d0 19 ca d0 f1 20 4c 19
02a8  4c ec 17 20 f3 19 cd e7
02b0  17 d0 08 20 f3 19 cd e8
02b8  17 f0 03 4c 29 19 6c e0
02c0  00 ad ed 17 85 e0 ad ee
02c8  17 85 e1 a9 91 8d f0 17
02d0  a9 17 8d f1 17 4c 91 17
02d8  00 00 00 4c 80 17 00 00
02e0  8d 00 00 4c c1 17 00 00
02e8  00 00 00 00 d0 ed
```

Dieses kurze KIM-Programm erzeugt den Autostart-Datenblock auf Band. Er wird dem zu ladenden Programm „vorgespannt", und dieses startet sich dann automatisch selbst

Üblicherweise steht dort 8D und in den beiden nachfolgenden Speicherstellen die Adresse, in die das vom Band eingelesene Byte eingeschrieben wird. Diese Adresse wird durch das Un-

Kaufen Sie keine Katze im Sack!

Niemand erwartet von Ihnen, 15.60 DM für etwas hinzublättern, dessen Inhalt Sie nicht kennen. Deshalb geben wir nachfolgend den Inhalt des soeben erschienenen FUNKSCHAU-Sonderheftes „Mikrocomputer-Anwendungen" wieder – zu Ihrer Information.

Basic

Autostart beim AIM-65-Basic. Das „Selread"-Verfahren: Virtuelle Mehrfachtabellen in Basic. Eingabe und Auslesen von Funktionen bei PET und CBM. Gewaltsamer Rücksprung aus Unterprogrammen: Der POP-Befehl. „Inflameter" fühlt der Mark auf den Puls. Kurztabellen sparen Speicherraum. Plotten mit dem CBM-Drucker. PET als Logikanalysator mit acht Kanälen. PET druckt Kalender. PET erzeugt DATA-Zeilen aus Maschinenprogrammen. PET hilft bei der Notenauswertung. PET tilgt Annuitäten. Der „Leerstring" in Microsoft-Basic. Witterungsgeführte Temperatur-Vorlaufregelung mit PET. Textbearbeitung mit dem TRS-80.

Hardware

AIM treibt Fernschreiber. Bootstrap-Reset für Z-80-Systeme. RAM-Erweiterung beim Eurocom-1. I/O-Testbox für das Lehr- und Lernsystem TM 990/189. Nascom-1-Verbesserungen. Serielle Schnittstelle für den TRS-80 mit dem VIA-Baustein 6522. Ziffernanzeige für den 8080.

Maschinensprache

8080/8085-Disassembler und Adressenrechner. Arithmetik-Routinen für den 8080. Berechnetes GOTO und RESTORE beim CBM. Display-Verzögerung beim AIM-65. DAA – ein Befehl mit Pfiff. Fernschreiber als AIM-Drucker. „Flagbild" hilft bei der Fehlersuche (6502). Formatierter Assembler-Ausdruck beim AIM-65. Komfort-Renumber für AIM-65-Basic. Nützliche AIM-Routinen. „Restore Line Number" beim AIM-65 und beim PET-2001. Single-Step beim AIM-Basic. Sorcerer steuert Fernschreiber. Sprachsynthese mit dem 6502. Z-80-Disassembler.

Taschenrechner

6 Gleichungen mit 6 Unbekannten (HP-29). Berechnung von Bessel-Funktionen mit dem HP-25. Sortierprogramm für den HP-41. TI-59 berechnet Entfernungen aus QTH-Kennern. TI-59-Symbolcode-Berechnung für alphanumerische Ausgabe.

ROM- und RAM-Tabellen

Ausführliche Dokumentation von RAM-Adressen und Unterprogrammen in Monitor- und Basic-ROMs bei den Computern PET-2001, CBM, KIM-1, AIM-65. Das Umschreiben von Programmen für Ihren eigenen Computer wird damit problemlos.

Außerdem finden sich im Sonderheft Mikrocomputer-Anwendungen die „Folgerungen aus Murphys Gesetz" – zum besseren Einschlafen nach langen Programmiernächten. Zitat daraus: „Die Wahrscheinlichkeit, daß ein Programm die ihm gestellten Anforderungen erfüllt, ist umgekehrt proportional dem Vertrauen des Programmierers in seine eigenen Fähigkeiten."

terprogramm „INCVEB" von Byte zu Byte laufend erhöht. Wenn die Adresse 17EC erreicht wird, kommt von der Kassette das Byte D0, also ein BNE-Befehl, und wird dort eingeschrieben. Die KIM-Lade-Routine liest das nächste Byte vom Band, erhöht mit INCVEB den Inhalt der Adresse 17ED von EC auf ED, und möchte das eben eingelesene Byte in den Speicher einschreiben. Das gelingt natürlich nicht, weil dort jetzt „D0 ED" eingespeichert ist. Selbstverständlich wird der relative Sprung ausgeführt, wenn das zuletzt gelesene Byte kein 00 war. Dafür muß gesorgt sein. Auf der anzuspringenden Adresse, das ist 17DB, wurde vorher bereits der absolute Sprung JMP 1780 geladen, das heißt, das soeben geladene Programm startet sich selbst und wartet auf den nächsten Datenblock, nämlich auf das gewünschte Programm, um dieses zu laden und selbst zu starten.

So ohne weiteres kann aber das ab 1780 geschriebene Programm nicht auf die Kassette gebracht werden, da auch die KIM-Save-Routine den VEB-Bereich mitbenützt und ein in 17EC gespeichertes Byte sofort verändern würde.

Folgende Prozedur hilft hier: Das Autostartprogramm wird z. B. auf 0280 geschrieben. Mit Hilfe des Hypertape-Programms von Jim Butterfield, das an zwei Stellen verändert wird, kann das ursprünglich ab 0280 stehende Programm auf der Kassette mit der Anfangsadresse 1780 versehen werden. Beim Hypertape mit der Startadresse $0100 wird bei 0123 statt „AD F5 17", die Byte-Folge „A9 80 EA" eingesetzt, und bei 0129 statt „AD F6 17" die Folge „A9 17 EA". Hypertape liest dann zwar das Programm ab $0280, setzt aber auf dem Band die Startadresse 1780 ein. Als Startadresse in 17F5 und 17F6 wird 0280 eingesetzt, die Endadresse in 17F7 und 17F8 ist 02EE.

Der Aufwand für das Erstellen eines Autostarts mag zunächst zwar groß erscheinen, ist aber im Vergleich zum Herstellen eines funktionierenden Programms sehr gering. Sind das selbstzustartende Programm und der Autostart-Teil erst einmal auf die Kassette gebannt, hat sich die kleine Mühe sicher gelohnt. *Peter Lyko*

Anm. der Red.: Ein Autostart-Programm für AIM-65 und PC-100 findet sich im FUNKSCHAU-Sonderheft „Programme für Kleincomputer und Taschenrechner". „Hypertape" ist z. B. in „Anwendungsbeispiele für den Mikroprozessor 6502", RPB 173, Franzis-Verlag, veröffentlicht.

Stundenpläne aus dem Computer

In einem Leserbrief in FUNKSCHAU 1980, Heft 13, wurde angefragt, ob es Programme für Kleincomputer gebe, mit denen die Erstellung von Stundenplänen für Schulen möglich ist.

Unsere Recherchen ergaben folgendes: Bisher existiert offensichtlich kein solches Programm, das auf einem Rechner mit z. B. 32 KByte RAM läuft, obwohl das vom Prinzip her durchaus machbar wäre.

Dagegen bietet die Hessische Zentrale für Datenverarbeitung (Mainzer Straße 29–33, 6200 Wiesbaden) Schulen in Hessen die kostenlose Stundenplanerstellung auf einem Großrechner (IBM 3033) an; zuständig ist der Leiter des Fachprojektes Bildungswesen, Herr Kossack.

Weiterhin existiert für die Siemens-Rechner 4004 bzw. 7000 mit den Betriebssystemen BS 1000 und BS 2000 ein Programm zur Stundenplanerstellung namens RIKS (Rang-Index-Kollisions-Strategie); es kann direkt von Siemens bezogen werden.

RIKS arbeitet sehr komfortabel, indem es zahlreiche Randbedingungen zuläßt, wie Lehrer-Verfügbarkeit, Kopplungen von Stunden unterschiedlicher Klassen, Fachraum-Verfügbarkeit, Doppelstunden-Wünsche, Stundenfixierung, Stundensperrung und Kurzzusammenfassung in Studienstufen. Erfahrungsgemäß führt die Lösung eines solch komplexen Problems auf rein mathematisch-analytischer Basis zu Schwierigkeiten; RIKS arbeitet daher durch versuchsweises Vertauschen von Stunden, Lehrern und Räumen, bis eine der meist mehreren möglichen Lösungen gefunden ist. Oft können allerdings nicht alle Stunden eingeordnet werden, weil sich u. U. Vorgabedaten widersprechen. Dann muß man von Hand noch etwas nachhelfen. Das Programm benötigt einen Rechner mit 128 KByte.

Es ist durchaus denkbar, daß es gelingt, den erforderlichen Speicherplatz zu vermindern, so daß eine Stundenplan-Erstellung auch auf einem Tischrechner mit weniger als 64 KByte RAM möglich wird. Fe.

Fakultät mit 3100 Stellen

Das Basic-Programm im *Bild* ist unverändert auf den Computern PET, CBM, PC-100 und AIM-65 lauffähig und berechnet die Fakultät einer Zahl bis zu einer nur durch den zur Verfügung stehenden Speicherplatz begrenzten Stellenzahl; beim 4-K-AIM z. B. bis zu 3100 Stellen. (Für Nicht-Mathematiker: Die Fakultät von z. B. 4 schreibt sich als 4! und errechnet sich zu 1 x 2 x 3 x 4. Einzige Ausnahme: 0! = 1.)

Da Zahlen im AIM-Basic maximal nur neun Stellen haben dürfen, werden sie in Gruppen von je fünf Ziffern aufgeteilt und in einer Matrix gespeichert. Die einzelnen Matrixwerte werden in den Zeilen 30 und 40 multipliziert; alle über 5 hinausgehenden Ergebnisstellen werden zum nächsthöheren Matrixwert addiert. Die Variable B zeigt dabei auf den aktuellen Matrixwert, während D die nötige Länge der Matrix angibt, die nötigenfalls in Zeile 60 verändert wird. Fakultäten bis 12 werden der Einfachheit halber in Zeile 110 berechnet.

Klaus-Gerd Meyer

```
10 INPUT"FAKULTAET VON";N:PRINT" ":IFN<13THENA=1:GOTO110
20 DIMA(620):A(0)=2:E=1E5
30 FORA=3TON:B=-1
40 B=B+1:C=A(B)*A+Z:Z=INT(C/E):A(B)=C-Z*E:IFB<=DTHEN40
50 IFZ=0THENNEXT:GOTO70
60 D=B:GOTO40
70 PRINTN"!="STR$(A(B));
80 FORA=DTO0STEP-1:A$=STR$(A(A)):A$=RIGHT$(A$,LEN(A$)-1)
90 PRINTRIGHT$("0000"+A$,5);:NEXT:PRINT" "
100 PRINT" ":PRINT"ZIFFERNANZAHL:";LEN(STR$(A(B)))-1+B*5:PRINT" ":END
110 FORB=1TON:A=A*B:NEXT:PRINTN"!="A:PRINT" ":END
```

Im Gegensatz zu den meisten Taschenrechnern stellt für dieses Basic-Programm auch die Fakultät von z. B. 150 noch lange kein Problem dar

Rudolf Hofer

BASIC vor dem Handbuch

4. Teil

Bild 14 zeigt das gesamte Programm zur Notenberechnung. Sie sollten es aber wirklich vorerst ohne zu spicken versuchen. Dieses Programm können Sie dann so abändern, daß mehr Namen und Fächer zugelassen sind. Eine etwas schwierigere Aufgabe ist es, das Programm so auszulegen, daß der Durchschnitt auch dann richtig berechnet wird, wenn eine Note fehlt. Versuchen Sie es, indem Sie an diese Stelle die Zahl Null in die DATA-Zeile setzen. Bei der Berechnung muß dann berücksichtigt werden, daß sich die Anzahl der Noten verringert.

Weitere Programme

Zur Übung sei das Studium einiger weiterer Programme empfohlen. In *Bild 15* finden Sie eines, das die Tagesdifferenz zwischen zwei Datumsangaben berechnet. Es erwartet vom Benutzer die Datumseingaben in der Form „Tag, Monat, Jahr". Wichtigster Bestandteil ist das Unterprogramm in *Bild 16*, in dem die Variable S berechnet wird. Die Variable gibt eine Anzahl von Tagen ab einem Bezugspunkt (T = 0, M = 0, J = 0) an. Das Programm wird zweimal aufgerufen, und die Differenz der beiden Ergebnisse wird in Zeile 200 ausgedruckt.

Wir stoßen hier zum erstenmal auf den Begriff „Unterprogramm". Dabei handelt es sich um eine Befehlsfolge, die mit GOSUB Zeilennummer (z.B. GOSUB 500) beliebig oft aus dem Hauptprogramm aufgerufen werden kann. Auf diese Weise spart man eine Menge Tipparbeit. Das wesentliche ist, daß nach dem Befehl RETURN, mit dem jedes Unterprogramm abgeschlossen werden muß, zu der dem Aufruf unmittelbar folgenden Zeile gesprungen wird. Das Unterprogramm in unserem Beispiel beginnt bei Zeile 500. Der Wert S

```
100 CLEAR 1000
110 DIM N(6,4)
120 DIM S$(6)
130 DIM F$(4)
200 REM EINGABE
210 FOR I=1 TO 6
220 READ S$(I)
230 NEXT I
250 FOR I=1 TO 4
260 READ F$(I)
270 NEXT I
280 FOR Z=1 TO 6
290 FOR S=1 TO 4
300 READ N(Z,S)
310 NEXT S
320 NEXT Z
350 REM SCHUELER ODER FACH
360 PRINT "NAME ODER FACH EINGEBEN";
370 INPUT B$
380 FOR Z=1 TO 6
390 IF S$(Z)=B$ THEN 600
400 NEXT Z
450 FOR S=1 TO 4
460 IF F$(S)=B$ THEN 700
470 NEXT S
480 PRINT "NICHT VORHANDEN"
490 GOTO 360
590 REM DURCHSCHNITT SCHUELER
600 G=0
620 FOR S=1 TO 4
630 G=G+N(Z,S)
640 NEXT S
650 PRINT "DER SCHUELER ";S$(Z);" HAT EINEN"
660 PRINT "NOTENDURCHSCHNITT VON ";G/4
690 REM FACHDURCHSCHNITT
695 GOTO 360
700 E=0
720 FOR Z=1 TO 6
730 E=E+N(Z,S)
740 NEXT Z
750 PRINT "DIE DURCHSCHNITTSNOTE IM FACH "
760 PRINT F$(S); " IST ";E/6
770 GOTO 360
1000 DATA AST,BAUM,DORN,FAUST,GROB,ZAUN
1010 DATA MATHEMATIK,PHYSIK,DEUTSCH,ENGLISCH
1020 DATA 2,3,3,1
1030 DATA 4,3,4,4
1040 DATA 1,2,1,3
1050 DATA 3,3,1,1
1060 DATA 5,4,6,4
1070 DATA 2,2,2,1
NAME ODER FACH EINGEBEN? BAUM
DER SCHUELER BAUM HAT EINEN
NOTENDURCHSCHNITT VON  3.75
NAME ODER FACH EINGEBEN? MATHEMATIK
DIE DURCHSCHNITTSNOTE IM FACH
MATHEMATIK IST  2.83333
NAME ODER FACH EINGEBEN? HANSELMEIER
NICHT VORHANDEN
NAME ODER FACH EINGEBEN
```

Bild 14. Gesamtprogramm zur Notenberechnung mit Berechnungsbeispiel

```
100 DIM M(12)
110 PRINT"ERSTES DATUM";
120 INPUT T,M,J
130 GOSUB 500
140 A=S
150 RESTORE
160 PRINT"ZWEITES DATUM";
170 INPUT T,M,J
180 GOSUB 500
190 B=S
200 PRINT"DIFFERENZ: ";B-A
210 RESTORE
220 GOTO 110
500 Z=0
510 IF J/4<>INT(J/4) THEN 540
520 IF M<=2 THEN 540
530 Z=1
540 Z=Z+(J-1)*365+INT((J-1)/4)
550 FOR K=1 TO 12
560 READ M(K)
570 NEXT K
580 S=Z+M(M)+T
590 RETURN
700 DATA 0,31,59,90,120,151,181,212,243,273,304,334
ERSTES DATUM? 1,1,1980
ZWEITES DATUM? 1,1,1981
DIFFERENZ:  366
```

Bild 15. Programm zur Berechnung der Differenz zwischen zwei Datumsangaben

setzt sich zusammen aus der Zahl der Tage in den vorhergehenden Jahren (Z) plus der Zahl der Tage im laufenden Jahr.

Z berechnet sich zu (J – 1) * 365 plus die Anzahl der zusätzlichen Tage in den Schaltjahren. Für diese Zusatztage brauchen wir die Integer-Funktion INT. Sie veranlaßt den Computer, aus einer beliebigen Zahl eine Ganzzahl zu machen, und zwar durch Abschneiden des Dezimalbruches. A = INT (2.75) hat z.B. zur Folge, daß A den Wert 2 erhält. INT ((J – 1)/4) gibt – wie leicht einzusehen ist – die Zahl der Schalttage (vom Bezugspunkt aus) an. Setzt man etwa als Jahr 1980 ein, dann sind vorher 1979 Jahre vergangen, von denen jedes vierte (wenn man die Ausnahmen an den Jahrhundertwenden außer Acht läßt) ein Schaltjahr war. 1979/4 ergibt aber 494,75; die Integer-Funktion macht daraus 494, die uns interessierende Zahl.

Nun fehlt noch die Zahl der Tage im laufenden Jahr. Sie setzt sich zusammen aus T (Monatstag) und der Zahl der in den Vormonaten vergangenen Tage M(M). Außerdem muß berücksichtigt werden, ob das laufende Jahr ein Schaltjahr ist. Die Abfrage daraufhin findet sich in Zeile 510. Ein Schaltjahr liegt vor, wenn die Jahreszahl ohne Rest durch vier teilbar ist, wenn also J/4 = INT (J/4) ist. In diesem Fall ist von Interesse, ob der Februar schon vorbei ist (Zeile 520). Wenn ja, wird Z auf einen Anfangswert von 1 gesetzt, der dem Schalttag entspricht.

In Zeile 540 wird – wie besprochen – die Tageszahl der vergangenen Jahre ermittelt, ergänzt um Z. Die Gesamtzahl wird wiederum mit Z bezeichnet. Die in den Vormonaten des interessierenden Datums vergangenen Tage ermitteln die Zeilen 550...570. Dies geschieht einfach, indem man die Werte der DATA-Zeile 700 einliest. M(M) hat dann den richtigen Wert: z.B. M(3) entspricht den bis März vergangenen Tagen (= 59). In Zeile 580 schließlich wird die Gesamtsumme S berechnet. Das Hauptprogramm ist eine Endlosschleife, die jeweils die Differenz der berechneten S-Werte ausdruckt. Gültig ist das Programm vom 1. März 1900 bis 28. Februar 2100.

In *Bild 17* ist ein Programm dargestellt, das den Wochentag zu einem Datum im selben Zeitraum ermittelt. Wie im vorhergehenden Fall wird eine Summe S in den Zeilen 500...580 berechnet, die der Tageszahl von einem Bezugspunkt an entspricht. Zieht man davon die größte durch sieben teilbare Zahl, die kleiner ist als S, ab (Zeile 590), dann erhält man einen Wert R zwischen 0 und 6, der unmittelbar dem Wochentag entspricht. Indem man R als Index für das zuvor eingelesene Feld T$(I) (mit Werten Samstag...Freitag) verwendet, kann man den Wochentag ausdrucken lassen. Mit welchem Wochentag das Feld beginnt, wurde an einem Beispiel ermittelt.

Zur Vertiefung sei das soeben im Franzis-Verlag erschienene Buch „Basic für Mikrocomputer" empfohlen; es beschreibt gängige Computer, deren Befehlssätze und zahlreiche Beispielprogramme (ISBN 3-7723-6821-2; 26 DM). ■

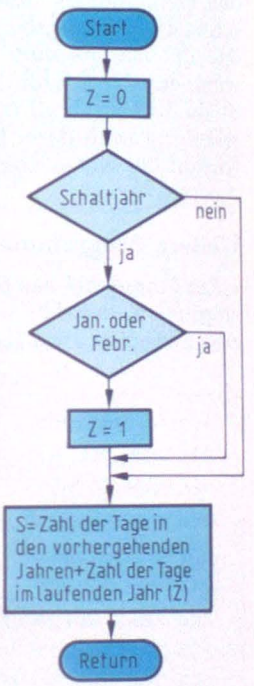


Bild 16. ▶ Flußdiagramm des Unterprogramms in Bild 15

▼ Bild 17. Ermittlung des Wochentags aus einer Datumsangabe

```
470 DIM M(12)
480 PRINT"TAG,MONAT,JAHR";
490 INPUT T,M,J
500 Z=0
510 IF J/4<>INT(J/4) THEN 540
520 IF M<=2 THEN 540
530 Z=1
540 Z=Z+(J-1)x365+INT((J-1)/4)
550 FOR K=1 TO 12
560 READ M(K)
570 NEXT K
580 S=Z+M(M)+T
590 R=S-INT(S/7)x7
600 FOR I=1 TO 7
610 READ T$(I)
620 NEXT I
630 PRINT T$(R+1)
700 DATA 0,31,59,90,120,151,181,212,243,273,304,334
710 DATA SAMSTAG,SONNTAG,MONTAG,DIENSTAG,MITTWOCH
720 DATA DONNERSTAG,FREITAG
TAG,MONAT,JAHR? 1,5,1980
DONNERSTAG
```

Hans-Georg Joepgen

Ein Basic ohne Nullprobleme

Betriebserfahrungen mit dem CP/A-Interpreter

Im Frühjahr 1980 wies die FUNKSCHAU darauf hin, daß sich in Kalifornien eine Gruppe von Außenseitern um ein neues Basic für die Maschinen der Apple-Klasse bemühe, das gewisse Schwächen des Microsoft-Interpreters überwinden sollte. Nun, der neue Interpreter ist auf dem Markt – zusammen mit einem neuen Disk-Operating-System und vielerlei anderen Dienstprogrammen ist er seit einigen Wochen im Rahmen des Software-Paketes „CP/A" per Benutzervertrag erhältlich. Der folgende Beitrag bringt erste Betriebserfahrungen.

Wie in [1] näher ausgeführt, können binär arbeitende Hochsprachen-Interpreter auch bei vergleichsweise geringen Anforderungen an die Anzahl der signifikanten Stellen aus prinzipiellen Gründen gewisse Dezimalzahlen nicht mit ausreichender Genauigkeit abbilden. Daß dies zu unerklärlichem Fehlverhalten von Programmen – besonders bei kaufmännischen Anwendungen – führt, die bisweilen nur durch lästige Korrekturroutinen zu meistern sind, davon wissen die Programmierer der TRS-80, PETs, PC-100, Basic-KIMs und Maschinen der Apple-Klasse ein Lied zu singen. *Bild 1* zeigt, wie liederlich die genannten Rechner bisweilen beispielsweise mit dem Wert Null umgehen – und um es gleich vorwegzunehmen: Mit CP/A-Business-Basic (so heißt der neue Interpreter offiziell) ist man derlei Ärger los (*Bild 2*). Allerdings – alles auf der Welt hat seinen Preis: So sehr ein im BCD-Code arbeitender Rechner mit seiner präzisen Zahlenverarbeitung im Bereich Mark, Pfennig und Prozent dem Kaufmann entgegenkommt, so deutlich treten dafür auf anderen Gebieten neue Schwächen auf. Bei technischen und wissenschaftlichen Anwendungen ist in gewissen Fällen eine rein binäre Abbildung offenbar günstiger: Zu seiner Überraschung erlebte der Verfasser nämlich, daß bei der Euler'schen Approximation eine Maschine 2020 unter CP/A bereits nach 393 Iterationen ins Schwindeln geriet, während es Computer mit einem Microsoft-Basic-Dialekt im Bauch anstandslos auf weit über 800 Schleifendurchläufe bringen [2]. Freilich, die CP/A-Verfasser betrachten ihren Interpreter nicht als Endprodukt; es ist daran gedacht, das gesamte Software-Paket kontinuierlich weiterzuentwickeln und den Vertrags-

```
]LIST
0  REM
1  REM FALLENTEST
2  REM ==========
100  PRINT  CHR$ (4)"PR#5"
110  FOR N =  - 1 TO 1 STEP .1
120  PRINT "          "N
130  NEXT : PRINT " "
140  PRINT "  ITT 2020 UNTER DOS
     3.2 (PALSOFT). -"

]RUN
          -1
          -.9
          -.8
          -.7
          -.6
          -.5
          -.4
          -.3
          -.2
          -.0999999997
          2.91038305E-10
           .1
           .2
           .3
           .4
           .5
           .6
           .7
           .800000001
           .900000001

 ITT 2020 UNTER DOS 3.2 (PALSOFT). -
```

Bild 1. Bis vor kurzem so etwas wie „Stand der Technik" bei den am häufigsten eingesetzten Basic-Interpretern: Unvollkommenheiten, wie sie hier zutage treten

```
0 REM
1 REM FALLENTEST
2 REM ==========
3 REM
100 OPEN #1,8,0,"P5:"
110 FOR N=-1 TO 1 STEP 0.1
120 PRINT #1,N
130 NEXT N:PRINT #1;"."
140 PRINT #1;"  ITT 2020 UNTER CP/A (BAS
IC V 2.0). -"
150 CLOSE #1

          -1
          -0.9
          -0.8
          -0.7
          -0.6
          -0.5
          -0.4
          -0.3
          -0.2
          -0.1
           0
           0.1
           0.2
           0.3
           0.4
           0.5
           0.6
           0.7
           0.8
           0.9
           1

 ITT 2020 UNTER CP/A (BASIC V 2.0). -
```

Bild 2. So dagegen benimmt sich ein neuer Interpreter für die Maschinen der Apple-Klasse. Seine Besonderheit: Interne Zahlenverarbeitung nicht länger binär, sondern im BCD-Code

```
10 OPEN #1,8,0,"P5:"
20 PRINT #1;"KOMMISSAR"
30 PRINT #1;"========="
40 PRINT #1;" ":PRINT #1;" "
50 N=0
60 ALTX=X:N=N+1
70 X=(1+1/N)^N:IF ALTX<X THEN 60
80 PRINT #1;" +++ ERSTER FEHLER BEI N =
";N
90 PRINT #1;" "
100 PRINT #1;"     UND                X =
 ";X;" . -"
110 PRINT #1;" "
120 PRINT #1;" +++ ABWEICHUNG  DELTA X =
 ";2.71828182-X
130 PRINT #1;" ":FOR N=1 TO 39:PRINT #1;
"-";:NEXT N:PRINT #1;"-"
140 CLOSE #1

KOMMISSAR
=========
 +++ ERSTER FEHLER BEI N = 393

    UND                X = 2.71482203 . -

 +++ ABWEICHUNG  DELTA X = 3.45979E-03

----------------------------------------
```

Bild 3. Hier vollzieht der Rechner nach, was zweihundert Jahre zuvor auf gleichem Wege dem Mathematiker Leonhard Euler gelang: Aufspüren der Zahl e. Eine harte Aufgabe für Computer, viele hundert Mal Potenzen mit dreistelligen Exponenten zu einer winzigen kleinen Grundzahl zu entwickeln

partnern zu mäßigem Preis mit den bisher geschriebenen Benutzerprogrammen vollständig kompatible Neuversionen ins Haus zu schicken. So kann man annehmen, daß CP/A wohl bald mehr Respekt vor dem Mathematiker Leonhard Euler zeigen wird, der dem Präzisionstest „Kommissar" den Algorithmus lieferte, mit dessen Hilfe die jetzt noch bestehende CP/A-Schwäche aufgedeckt wurde (*Bild 3*).

Eine erste Aktualisierungs-Diskette ist unterdessen an die Vertragskunden ausgeliefert: Die Basic-Neuversion V 2.0 brachte auch gleich neben anderen Annehmlichkeiten ein hochkomfortables PRINT USING. Es ist beim Präzisionstest „Mercator" gleich mit eingesetzt, mit dessen Hilfe die Kommerz-Tauglichkeit von CP/A untersucht wurde (*Bild 4*). Da das Listing einige Eigentümlichkeiten des neuen Basic-Dialektes erkennen läßt, soll das Programm hier näher besprochen werden.

„Mercator" zeigt Charakteristisches auf

In Zeile 10 wird ein Datenkanal zu jenem Peripheriegerät geschaltet, das über den Steckplatz Nummer 5 des Computers angeschlossen wurde, hier ein Drucker. Alle Print-Befehle mit nachgesetzter Datenkanal-Nummer geben dorthin aus, solche ohne Nummer weiterhin auf den Schirm. Die Standard-Software von ITT 2020 und Apple erlaubt dergleichen nicht, da muß – weil gleichzeitig immer nur ein Ausgabekanal zur Verfügung steht – laufend umgeschaltet werden. Ein Punkt für CP/A also, aber schon setzt's einen Minus-Punkt: Strings müssen vor Benutzung dimensioniert werden, es wird nicht einmal, wie auf dem neuen HP 85, eine Ersatzdimensionierung vorgenommen. Bei dieser Gelegenheit sei auch gleich erwähnt, daß sich CP/A eng an die Hewlett-Packard-Syntax bei der Stringverarbeitung anlehnt und auch ähnliche Einschränkungen beim Errichten von Datenfeldern wie der HP-85 zeigt [3; 4].

Nun setzt's wieder einen Punkt für CP/A: Variable dürfen, gleichgültig, ob numerisch oder auf Zeichenketten bezogen, auf Skalare oder Felder, einen vollen Namen tragen, (fast) beliebig viele Zeichen lang. Besonders Lehrer werden dies zu schätzen wissen, denn

```
0 REM ************************
1 REM *                      *
2 REM *   M E R C A T O R    *
3 REM *   ===============    *
4 REM *                      *
5 REM *  (JOEPGEN/13.7.80)   *
6 REM *   PRAEZISIONSTEST    *
7 REM * FUER KAUFM. ANWEND.  *
8 REM *                      *
9 REM ************************


10 OPEN #1,8,0,"P5:":DIM MASK$(24):MASK$
="DM ###/ ###/ ###/ ###.&&"
20 PRINT #1;"MERCATOR"
30 PRINT #1;"========="
40 PRINT #1;" ":GOSUB 240:PRINT #1;" "
50 N=0:GRUNDBETRAG=10000000
60 N=N+1:ALTBETRAG=GRUNDBETRAG
70 PRINT  USING MASK$,GRUNDBETRAG
80 GRUNDBETRAG=GRUNDBETRAG+GRUNDBETRAG*1
.0E-03:REM ERHOEHUNG UM 1 PROMILLE
90 PRUEFBETRAG=GRUNDBETRAG+0.01
100 IF PRUEFBETRAG>GRUNDBETRAG THEN 60
110 PRINT #1;" +++ DER ERSTE PFENNIG GIN
G VERLOREN"
120 PRINT #1;"     BEI EINEM GRUNDBETRAG
 VON":PRINT #1;" "
130 PRINT #1; USING MASK$,GRUNDBETRAG
140 PRINT #1;" "
150 GOSUB 240
160 PRINT #1;" +++ VORANGEGANGENER GRUND
BETRAG:":PRINT #1;" "
170 PRINT #1; USING MASK$,ALTBETRAG
180 PRINT #1;" "
190 GOSUB 240
200 PRINT #1;" ":PRINT #1;"     KONTROLL
ZAEHLER   N: ";N
210 GOSUB 240
220 PRINT #1;" "
230 CLOSE #1:PRINT " +++ PROGRAMM BEENDE
T.-":PRINT CHR$(7):END
240 FOR N=1 TO 39:PRINT #1;"-";:NEXT N:P
RINT #1;"-":RETURN


MERCATOR
========

----------------------------------------

 +++ DER ERSTE PFENNIG GING VERLOREN
     BEI EINEM GRUNDBETRAG VON

DM      100 026 326.00

----------------------------------------
 +++ VORANGEGANGENER GRUNDBETRAG:

DM       99 926 400.01

----------------------------------------

      KONTROLLZAEHLER   N: 40
----------------------------------------
```

Bild 4. Ein Beispiel, an dem die Überlegenheit von BCD-Interpretern gegenüber dem binären Microsoft-Basic schön sichtbar wird: Kein Wunder, der Zahlenvorrat des neuen Interpreters reicht auch zehnstellig von 10^{-99} bis 10^{99}

nichts erschwert das Erlernen einer Computersprache so sehr wie der Abkürzungs-Dschungel, der entsteht, wenn von Namen nur die ersten zwei Zeichen relevant sind [5] wie in den Microsoft-Basicdialekten. Dimensioniert und gleich mit merkwürdigen Zeichen gefüllt wurde die Stringvariable MASK. Um ihre Funktion klarzumachen, springen wir gleich nach Zeile 130. Sie schickt den Inhalt von Grundbetrag an den Drucker, und zwar (PRINT ...USING) in einer durch den Inhalt von MASK bestimmten Form: Die Abkürzung für Deutsche Mark vorndran, die Tausenderkolonnen getrennt und auf jeden Fall zwei Stellen rechts vom Komma, auch, wenn's eigentlich gar keine Pfennige gibt. Als letzte Besonderheit gegenüber Dialekten wie Palsoft und Applesoft ist zu verzeichnen, daß unser Datenkanal 1 nach Benutzung wieder durch CLOSE aufgelöst werden muß; für PET-Benutzer klingt dies vertraut.

Zehn Millionen plus ein Pfennig

Was macht nun „Mercator" im Einzelnen? Das Programm nimmt das schöne Sümmchen von 10 Millionen Mark in die Variable GRUNDBETRAG, dann nochmals gleich viel Geld in die Variable PRUEFBETRAG, und addiert schließlich noch einen bescheidenen Pfennig zu PRUEFBETRAG. Sodann wird geprüft, ob CP/A-Basic den Inhalt beider Variablen noch unterscheiden kann. Erkennt der Interpreter den feinen Unterschied, dann gibt es Tageszinsen auf die Millionen in GRUNDBETRAG: $^1/_{10}$ Prozent (die Bankenaufsichts-Behörde möge derlei Wucherzinsen verzeihen); das Spiel mit dem Pfennig beginnt von neuem. Kann der Computer den Pfennig nicht mehr würdigen, gibt er auf und meldet sich: der Probelauf zeigt, wie und wann. Eine Aufgabenstellung, mit der sich PALSOFT von Anfang an überfordert zeigte, wie *Bild 5* zeigt: Zwar kann der Interpreter die beiden Zahlen anfänglich beim Größenvergleich im Zehn-Millionen-Bereich noch mit Mühe auseinanderhalten, beim Ausdrucken verschwindet der Pfennig jedoch offenbar schon in einer Schutzstelle; beim Wechsel in die nächsthöhere Größenordnung ist dann endgültig Feierabend. Mit Mercator wird übrigens nur untersucht, wann Unsicherheiten der letzten Stelle relevant werden.

Nicht geprüft wurde, ob die fortgesetzte Multiplikation nicht ihrerseits Fehler ins Programm schleppt. Dies zu überprüfen, wäre vielleicht eine reizvolle Aufgabe für einen FUNKSCHAU-Leser mit Zugang zu einem größeren Computer (oder präzisem Taschenrechner!). Eines jedoch ist nachgewiesen: Nicht nur bei der Meisterung von Null-Problemen, auch bei kaufmännischen Berechnungen schlechthin ist der CP/A-Interpreter den Microsoft-Vorgängern überlegen: Höhere Präzision plus PRINT USING machen die Maschinen der Apple-Klasse eigentlich erst zum seriösen Gerät. Zu einer kritischen Würdigung gehört freilich auch der Hinweis, daß der Verfasser für Aufgaben mit größeren Text-Verarbeitungs-Aktionen dennoch allemal TRS-80 Level II, Commodore-Basic, Applesoft und Palsoft dem Neuling auf dem Interpreter-Markt vorzieht: Dies um so mehr, als all die wunderschönen Möglichkeiten zur Gestaltung farbiger Grafiken dem CP/A-Benutzer bis zur besprochenen Version 2.0 nicht zugänglich sind.

Editor, Assembler und Debugger

Business Basic, dem vor allem diese Besprechung gilt, ist nicht mit dem Control Program/Apple (CP/A) allein im Software-Paket. Wiewohl diese Kombination zu geringerer Lizenzgebühr auch allein zu haben ist, empfiehlt es sich, den Disk File Manager DFM und vor allem EASMD mitzubestellen: Einen komfortablen Editor, einen wunderschönen Assembler mit Möglichkeiten, die schon zum Schwärmen verleiten könnten, und eine „Debugger" genannte Erweiterung zum ohnehin schon exzellenten Monitor der Apple- und 2020-Versionen: Dies alles unter einem dem berühmten CP/M deutlich nachempfundenen Betriebssystem mit vielerlei Vorzügen (und einigen Nachteilen) gegenüber DOS 3.2 und DOS 3.21. Hierüber wird vielleicht später noch gesondert zu berichten sein; zuvor galt des Verfassers Aufmerksamkeit und Erprobungs-Lust aus gutem Grund erst einmal hauptsächlich dem neuen Basic, denn dafür gibt's sicherlich den größeren Freundeskreis.

Literatur

[1] *Joepgen, Hans-Georg:* Vorsicht – Falle! Die „Nullprobleme" binärer Basic-Interpreter. FUNKSCHAU 1980, Heft 2, S. 79, Franzis, München.

[2] *Joepgen, Hans-Georg:* „Kommissar" deckt Rechenungenauigkeit auf. „Hobby-Computer 1", Sonderheft der Zeitschriften ELO, FUNKSCHAU und ELEKTRONIK, S. 96, Franzis, München.

[3] *Valentin, Johannes:* HP-85: Kommerzielle Qualität und viel Komfort. ELEKTRONIK 1980, Heft 2, Franzis, München.

[4] *Feichtinger, Herwig:* Das Capricorn-Projekt. FUNKSCHAU 1980, Heft 3, Franzis, München.

[5] *Joepgen, Hans-Georg:* „Variablen-Wächter" sorgt für mehr Programm-Transparenz. ELEKTRONIK 1980, Heft 8, Franzis, München.

[6] *Optimized Systems Software (Herausgeber):* CP/A Control Program Apple – OSS Business Basic. Shepardson Microsystems, Inc. (20395 Pacifica Dr., S. 108), Cupertino, CA 95014, USA.

Stichworte zum Inhalt

Apple-II, CP/A, Business Basic, Rechengenauigkeit, Microsoft-Basic.


```
]LIST
10  REM PFENNIGTEST
20  REM ===========
30  REM JOE/13.7.80
40  REM
50 A = 10000000
60 B = 10000000.01
70  PRINT " "
72  PRINT "FUER MICH GILT:"
73  PRINT "---------------"
80  PRINT "A:    "A" DM   B:    "B" DM"
90  PRINT " "
100  PRINT : IF A = B THEN  PRINT "
     +++ ICH BIN DOCH KEIN PFENNIGFUCHSER. "
     : GOTO 120
110  PRINT " +++ ICH EHRE DEN PFENNIG. -"
120  FOR N = 1 TO 38: PRINT "-";: NEXT
```

```
]RUN

FUER MICH GILT:
---------------
A:    10000000 DM   B:    10000000 DM

 +++ ICH EHRE DEN PFENNIG. -
----------------------------------------

50 A = 100000000
60 B = 100000000.01
]RUN

FUER MICH GILT:
---------------
A:    100000000 DM   B:    100000000 DM

 +++ ICH BIN DOCH KEIN PFENNIGFUCHSER.
----------------------------------------
```

Bild 5. Wo CP/A-Basic noch souverän den letzten Pfennig handhabt, werden Microsoft-Interpreter neurotisch: Im Beispiel rechts oben zeigt sich eine ITT 2020 als schizophren: Einerseits weiß sie noch um den Unterschied zwischen den Zahlen A und B, druckt sie aber andererseits dennoch mit gleichen Werten. Im Beispiel rechts unten ist dann der Ofen endgültig aus.

„Richtige" Kleinschreibung beim PET

Wie bei manchen anderen Mikrocomputern werden beim PET-2001 Großbuchstaben erzeugt, wenn die Shift-Taste nicht gedrückt ist; drückt man sie, so entstehen Kleinbuchstaben. Dieses Verhalten ist also genau umgekehrt gegenüber einer Schreibmaschine.

```
03A5:   A4 E2       LDY   $E2
03A7:   B1 E0       LDA   ($E0),Y
03A9:   8D 27 80    STA   $8027
03AC:   AD 0F 02    LDA   $020F
03AF:   C9 03       CMP   =$03
03B1:   D0 0A       BNE   $03BD
03B3:   AA          TAX
03B4:   AD FF 03    LDA   $03FF
03B7:   49 80       EOR   =$80
03B9:   8D FF 03    STA   $03FF
03BC:   8A          TXA
03BD:   AE FF 03    LDX   $03FF
03C0:   E8          INX
03C1:   30 16       BMI   $03D9
03C3:   C9 40       CMP   =$40
03C5:   90 12       BCC   $03D9
03C7:   C9 60       CMP   =$60
03C9:   B0 04       BCS   $03CF
03CB:   69 80       ADC   =$80
03CD:   90 0A       BCC   $03D9
03CF:   C9 C0       CMP   =$C0
03D1:   90 06       BCC   $03D9
03D3:   C9 E0       CMP   =$E0
03D5:   B0 02       BCS   $03D9
03D7:   E9 7F       SBC   =$7F
03D9:   8D 0F 02    STA   $020F
03DC:   4C 7E E6    JMP   $E67E
03DF:   78          SEI
03E0:   AD 19 02    LDA   $0219
03E3:   49 77       EOR   =$77
03E5:   8D 19 02    STA   $0219
03E8:   AD 1A 02    LDA   $021A
03EB:   49 E5       EOR   =$E5
03ED:   8D 1A 02    STA   $021A
03F0:   58          CLI
03F1:   60          RTS
03F2:   A9 03       LDA   =$03
03F4:   48          PHA
03F5:   A9 A5       LDA   =$A5
03F7:   48          PHA
03F8:   08          PHP
03F9:   08          PHP
03FA:   08          PHP
03FB:   08          PHP
03FC:   4C 85 E6    JMP   $E685
03FF:   00          !!!
```

Wie eine gewöhnliche Schreibmaschine läßt sich der PET mit dem dazugehörigen Drucker CBM 3023 verwenden, wenn man dieses Programm bei 03A5 startet

Mit dem Programm im Bild können Sie wieder gewohnte Verhältnisse herstellen. Es wird durch SYS 991 aufgerufen, wobei als Erinnerung ein zweiter Cursor am linken Bildschirmrand erscheint (durch LDA # SAC in Zeile 03F5 entfällt der zweite Cursor). Da das Programm den IRQ-Vektor benutzt, können Sie nun wie gewohnt Programme schreiben und ausführen, wobei Sie nun bei Kleinschreibung bzw. bei Graphiksymbolen nicht mehr fortwährend die Shift-Taste drücken müssen. Wenn Sie ein Programm schreiben, genügt ein einmaliges Drücken der BREAK-Taste, um auf normale bzw. inverse Tastenfunktion zu wechseln. Aus einem Programm heraus kann die Umschaltung natürlich nicht mehr durch die BREAK-Taste erfolgen. Man muß dann entweder in Zeile 03AF eine andere Taste definieren oder aber direkt durch die POKE 1023,0 bzw. POKE 1023,128 umschalten.

Die einzige Einschränkung ist, daß bei Kassettenoperationen das Programm zuvor (durch erneutes SYS 991) ausgeschaltet werden muß. Man kann es anschließend wieder einschalten, was auch aus einem laufenden Programm heraus erfolgen kann.

Beim PET wird die Tastatur 60mal pro Sekunde durch einen Hardware-Interrupt abgefragt, wobei der gegenwärtige Stand des Programmzählers und der Prozessorstatus auf den Stack gelegt werden. Dann holt der Prozessor aus den Speicherzellen FFFE-FFFF (ROM) die Adresse für den Einstieg in das Programm zur Tastaturabfrage. Hier werden zu Beginn die Registerinhalte (Akku, X, Y-Register) gerettet (auf den Stack gelegt), bevor das eigentliche Abfrageprogramm über die Adresse in 0219-021A (E685) angesprungen wird.

Da dieser Vektor im RAM-Bereich liegt, kann er verändert werden, d. h. ein Programm kann vor der T.-Abfrage eingefügt werden. Da die Tastatur-Routine jedoch mit einem Return from Interrupt (RTI) endet, gibt es leider keine direkte Möglichkeit, ein Programm *nach* der T.-Abfrage einzufügen, um die Eingaben zu verändern.

Man kann jedoch vor dem Sprung zur Abfrage eine neue RTI-Adresse auf den Stack schieben, so daß das Ende des Interrupts nicht zu einem Rücksprung in das unterbrochene Hauptprogramm führt, sondern zu einem Korrekturprogramm, wo die Eingaben (Groß-, Kleinbuchstaben) umgedreht werden. Hier muß sich natürlich der korrekte Rücksprung in das Hauptprogramm anschließen.

Heino Velder

Römische Zahlen – spanische Dörfer?

Das im *Bild* aufgelistete PC-100/AIM-65-Programm ermittelt die römische Schreibweise einer Dezimalzahl und bedarf wohl keines weiteren Kommentars. Erwähnt sei lediglich, daß der zuweilen auftauchende Befehl „PRINT" lediglich einen Zeilenvorschub bewirkt. Kleine Fleißaufgabe für unsere Leser: Versuchen Sie, die Zeilenzahl des Programm zu reduzieren!

Holger Scholz

```
LIST
  5 A$=""
 10 INPUTA
 15 A=INT(A)
 20 IFA>3888THEN10
 23 IFA<1THEN10
 25 B=A
 30 IFA<1000THEN70
 40 A$=A$+"M"
 50 A=A-1000
 60 GOTO30
 70 IFA<500THEN100
 75 IFA>899THEN360
 80 A$=A$+"D"
 90 A=A-500
100 IFA>399THEN150
110 IFA<100THEN170
120 A$=A$+"C"
130 A=A-100
140 GOTO110
150 A$=A$+"CD"
160 A=A-400
170 IFA<50 THEN200
175 IFA>89THEN380
180 A$=A$+"L"
190 A=A-50
200 IFA<40THEN240
210 A$=A$+"XL"
220 A=A-40
230 GOTO280
240 IFA<10THEN280
250 A$=A$+"X"
260 A=A-10
270 GOTO240
280 IFA=9THEN400
290 IFA<5THEN320
300 A$=A$+"V"
310 A=A-5
320 IFA=0THEN440
325 IFA=4THEN420
330 A$=A$+"I"
340 A=A-1
350 GOTO320
360 A$=A$+"CM"
370 A=A-900:GOTO170
380 A$=A$+"XC"
390 A=A-90:GOTO280
410 A$=A$+"IX":GOTO
440
420 A$=A$+"IV"
440 PRINTB;" ";A$
450 GETA$:IFA$=""TH
EN450
460 GOTO5
470 END
```

Zur Umwandlung dezimaler in römische Zahlen dient dieses kleine Basic-Programm. Es läuft unverändert auf den meisten Personal-Computern (AIM-65, PC-100, PET, CBM usw.)

Schnelleres Suchen beim Texteditor

In Heft 22/1980 fanden sich sowohl eine verbesserte Version des CBM-Texteditors, dessen ursprüngliche Version in Heft 1/1980 [1] abgedruckt war, als auch die CBM-Implementierung der Instring-Funktion, die eine sehr schnelle Stichwortsuche ermöglicht. Was liegt näher, als beides zu kombinieren?

Die nötigen Änderungen des Texteditor-Programms aus Heft 22 [2] sind deshalb in *Bild 1* wiedergegeben. Dabei muß natürlich die Instring-Funktion durch Laden des Maschinenprogramms [3] vorher initialisiert werden. Um dem Benutzer keine unnötigen Komplikationen zu bereiten, wurde dabei folgender Weg gewählt:

Unter dem Filenamen „e" (wie „Editor") wird zunächst das in *Bild 2* abgedruckte Initialisierungsprogramm auf Diskette abgespeichert. Dann zeichnet man den geänderten Editor (siehe Bild 1) auf Diskette mit dem Filenamen „ed" auf.

Lädt man später nun „e" von der Diskette und gibt RUN ein, so wird automatisch die Instring-Funktion initialisiert; bis das geschehen ist, erscheint auf dem Bildschirm „Bitte warten". Dann lädt das Initialisierungsprogramm ohne Zutun des Benutzers von der Diskette das Texteditorprogramm „ed" und startet es. All dies erfolgt mit einem kleinen Trick: Die hierzu nötigen Ladebefehle und RUN werden auf den Bildschirm geschrieben und automatisch in den Tastaturpuffer des CBM übernommen, so daß sie auch ausgeführt werden.

Die Inbetriebnahme des Texteditors inklusive der Instring-Funktion erfordert deshalb nur folgende Eingaben:
OPEN 1, 8, 15, "I1"
(Diskette links)
LOAD "e", 8
RUN

Alles Weitere erfolgt automatisch, und nach kurzer Zeit meldet sich das Editor-Programm mit der Zeilenbreite-Abfrage. *Herwig Feichtinger*

Literatur

[1] Basic-Texteditor. FUNKSCHAU 1980, Heft 1.
[2] Basic-Texteditor. FUNKSCHAU 1980, Heft 22.
[3] *Martin, Reinhold:* Schnelle Stichwortsuche beim CBM. FUNKSCHAU 1980, Heft 22.

Kompatible Grafiken beim Superboard

Das Superboard II ist für Graphik-Spiele gut geeignet, da der Zeichensatz eine Reihe von Symbolen wie Flugzeuge, Panzer etc. enthält. Da das Video-Interface aber nur 24 x 24 Zeichen auf den Bildschirm bringt, existieren verschiedene Umbauvorschläge, von 16 x 48 (beim englischen „Bruder", dem UK101) bis zu 30 x 50 cm. Wenn nun Graphik-Programme auf einem anderen Superboard laufen sollen, so gerät unweigerlich das Bild durcheinander, d.h., man sitzt mehrere Stunden vor der Maschine und muß das ganze Programm durchgehen, um das Bild wieder in Ordnung zu bringen.

Dabei könnte man das Problem durch Einführen von 4 Variablen sehr einfach lösen. Man definiert sie am Anfang des Programms, etwa folgendermaßen:

```
1 OL=53316 : REM OBERE LINKE ECKE (des sichtbaren Video-RAMs)
2 ZL=32 : REM ZEILENLAENGE
3 BL=24 : REM BILDLAENGE
(d. h., Anzahl der sichtbaren Zeichen)
4 AZ=24 : REM ANZAHL DER (sichtbaren) ZEILEN
```

Nun berechnet man die Positionen, die man (mit PEEK oder POKE) ansprechen will, aus diesen Variablen. Die Bildmitte z.B. ergibt sich

BM=OL + INT(AZ/2) x ZL + INT(BL/2)

Es empfiehlt sich die reichliche Verwendung der INT-Funktion. Wird das Programm getauscht, braucht der Empfänger nur die ersten Zeilen zu kontrollieren und ggf. zu verändern, und schon stimmt das Bild. *Axel Unterschuetz*

```
200 J=-1
290 INPUT"sUCHBEGRIFF=";B$:K=LEN(B$)
300 J=J+1:IFA$(J)="END"THENPRINT:RETURN
310 M$=A$(J)
315 S=@INST(M$,B$)
320 IFS=0THEN300
330 PRINTM$
340 FORS=0TO1000:NEXT
362 GETE$:IFE$<>" "GOTO300
364 PRINT"RF=fORTSETZUNG,S=sTOPr ";
366 GETE$:IFE$="S"THENPRINT"stop":RETURN
368 IFE$="F"THENPRINT" ok":GOTO300
369 GOTO366
380 I=-1:REM INSERT
```

Bild 1. Geänderte Zeilen beim Texteditor-Programm aus Heft 22. Die Instring-Funktion vereinfacht nicht nur das Programm, sondern erhöht auch die Geschwindigkeit der Stichwort-Suche. Ein Unterbrechen der Suche ist jederzeit mit der Leertaste möglich

```
450 PRINT"sbITTE WARTEN
460 X=7821:Y=PEEK(53)-32:X=256*Y+X
470 FOR I=0 TO 370
480 READ A:IFA<0THENA=Y-A
490 POKE (X+I),A
500 NEXT I
510 SYS(7821+256*Y)
520 POKE158,2:POKE623,13:POKE624,13
525 PRINT"sQQLOAD"CHR$(34)"ED"CHR$(34)",8":PRINT"QQQQRUN":PRINT"S";
570 DATA 169,141,133,052,169,-30,133,053,169,076,133,121,169,162,133,122,169
580 DATA -30,133,123,096,201,064,240,036,201,058,176,247,076,125,000,230,119
590 DATA 208,002,230,120,096,165,119,208,002,198,120,198,119,096,032,173,-30
600 DATA 208,003,032,180,-30,160,000,177,119,096,164,120,192,002,240,214,169
610 DATA 000,133,177,133,078,032,194,-30,201,178,240,003,076,003,206,032,142
620 DATA 204,165,008,072,165,068,133,087,165,069,133,088,032,173,-30,160,005
630 DATA 032,112,000,217,242,-31,208,226,136,208,245,032,189,-30,032,109,207
640 DATA 032,144,204,032,223,-31,160,000,177,068,133,080,200,177,068,141,122
650 DATA -31,141,157,-31,200,177,068,141,123,-31,141,158,-31,032,248,205,032
660 DATA 189,207,032,144,204,032,223,-31,160,000,177,068,133,083,200,177,068
670 DATA 141,119,-31,141,152,-31,200,177,068,141,120,-31,141,153,-31,032,197
680 DATA -30,201,041,240,039,201,044,240,003,076,003,206,032,173,-30,032,159
690 DATA 204,032,142,204,032,154,208,165,098,133,177,197,080,144,002,176,060
700 DATA 032,197,-30,201,041,240,003,076,003,206,166,177,173,048,004,221,034
710 DATA 004,240,009,232,228,080,208,243,162,000,240,029,134,079,169,000,133
720 DATA 089,230,079,230,089,164,089,196,083,240,012,185,048,004,164,079,217
730 DATA 034,004,208,221,240,234,232,134,078,165,087,133,068,165,088,133,069
740 DATA 104,016,014,160,000,169,000,145,068,200,165,078,145,068,076,212,-31
750 DATA 169,000,133,095,164,078,132,096,162,144,056,032,085,219,166,068,164
760 DATA 069,032,227,218,104,104,104,104,032,173,-30,076,196,198,096,224,000
770 DATA 208,251,162,008,189,247,-31,032,210,255,202,208,247,160,040,076,125
780 DATA 245,040,084,083,078,073,071,078,073,082,084,083,063,013,141,133,052
```

Bild 2. ▶
Initialisierungsprogramm. Es initialisiert die Instring-Funktion, lädt automatisch das Editorprogramm von der Diskette nach und startet es . Man beachte, daß der verwendete Drukker Groß- und Kleinbuchstaben vertauschte: die unterstrichenen Steuerzeichen sind: Q = Cursor nach unten, s = Bildschirm löschen, S = Cursor home, R = reverse, r = reverse off

Hoch- und Tiefpaß-Berechnung in Basic

Das hier vorgestellte Programm berechnet die Anzahl und die Werte der Elemente für die gewählte Betriebsfrequenz, Sperrfrequenz und Dämpfung bei der Sperrfrequenz. Da die Anzahl der Elemente aufgerundet wird, ist die Filterdämpfung bei der Sperrfrequenz etwas größer als die gewählte Dämpfung. Eingabefehler werden teilweise durch das Programm erkannt und es wird zur erneuten Eingabe der Filterwerte aufgefordert.

Die ermittelten Werte lassen sich mit dem Programm „Passive Ketten-Netzwerke" aus dem Funkschau-Sonderheft „Hobby-Computer 1" sehr gut überprüfen.

Bild 1 zeigt das Schaltbild eines Butterworth-Tiefpaß-Filters, *Bild 2* den dazugehörigen Frequenzgang. Die *Bilder 3* und *4* geben Entsprechendes für einen Butterworth-Hochpaß wieder. In *Bild 5* ist das Basic-Programm in einer Version für den PET-2001 abgedruckt.

Zum Schluß noch ein kleines Rechenbeispiel: *Bild 6* zeigt die Schaltung eines Tiefpaßfilters, das aus neun Elementen besteht. Gesucht ist die Dimensionierung für eine 3-dB-Grenzfrequenz von 35 MHz (f_c) und eine 80-dB-Sperrfrequenz von 100 MHz (f_x). Das Programm errechnet aus diesen Ausgangsdaten folgende Bauelemente:

C1 =	31,58 pF	L6 =	427,3 nH
L2 =	227,4 nH	C7 =	139,3 pF
C3 =	139,3 pF	L8 =	227,4 nH
L4 =	427,3 nH	C9 =	31,58 pF
C5 =	181,9 pF		

Eduard Nolte

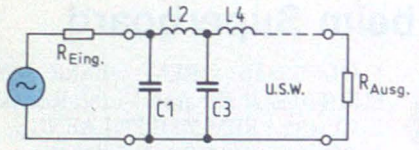


Bild 1. Prinzipschaltbild eines passiven Tiefpaßfilters in Pi-Anordnung

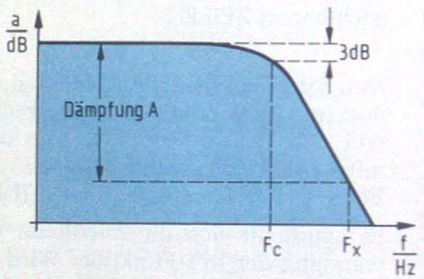


Bild 2. Amplitudenfrequenzgang eines Butterworth-Tiefpaßfilters

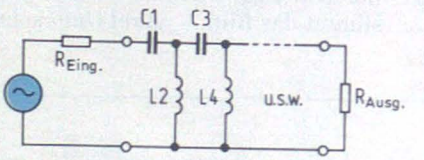


Bild 3. Passives Hochpaßfilter in Pi-Anordnung

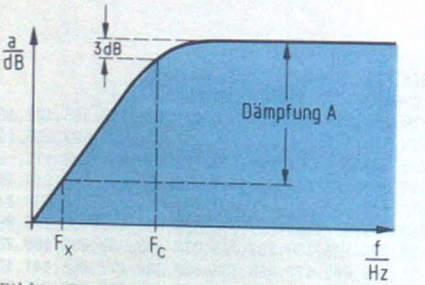


Bild 4. Frequenzgang eines Butterworth-Hochpaßfilters

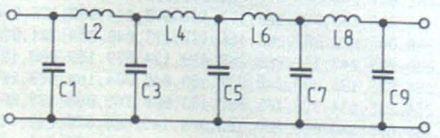


Bild 6. Schaltung eines viergliedrigen Tiefpaßfilters, wie es zur Oberwellenunterdrückung von Kurzwellensendern eingesetzt wird

```
100 REM BERECHNET BUTTERWORTH TIEFPASS
110 REM UND HOCHPASS FILTER
120 REM
130 REM COPYRIGHT BY E. NOLTE 1980
140 REM
150 PRINT:PRINT"EINGABE EING./AUSG.WIDERSTAND"
160 INPUT "R IN OHM =";R
170 PRINT
180 PRINT"EINGABE TIEFPASS=TP  ODER  HOCHPASS=HP
190 INPUT EN$
200 PRINT
210 PRINT"EINGABE ECKFREQUENZ (-3DB) FC"
220 INPUT"FC IN HERTZ =";FC
230 PRINT
240 PRINT"EINGABE SPERRFREQUENZ FX"
250 INPUT"FX IN HERTZ =";FX
260 PRINT
270 PRINT"EINGABE DAEMPFUNG A IN DB BEI FX"
280 INPUT"A IN DB =";A
290 PRINT
300 IFEN$="HP"THEN330
310 IFEN$="TP"THEN350
320 PRINT"EINGABEFEHLER"  :  GOTO 180
330 IF FX/FC>1 THENPRINT"EINGABEFEHLER/
    FC UND FX VERTAUSCHT" :GOTO 180
340 N=A/(20*0.4343*LOG(FC/FX)) :GOTO370
350 IF FC/FX>1 THEN PRINT"EINGABEFEHLER/
    FC UND FX VERTAUSCHT" :GOTO 180
360 N=A/(20*0.4343*LOG(FX/FC))
370 N=INT(N+1)
380 DIM B(N)
390 PRINT
400 PRINT"N =";N
410 PRINT
420 FORI=NTO1 STEP-1
430 B(I)=(2*I-1)*3.14159/(2*N)
440 NEXT
450 B=1/(3.14159*R*FC) :C=-2
460 FORI=1TON
470 I$=STR$(I)
480 PRINT I$;"    ";
490 GOSUB560
500 PRINT TAB(6);CL$ ;
510 IFEN$="TP"THEN580
520 IFEN$="HP"THEN620
530 C=C*(-1):RR=R↑C:B=B+RR
540 NEXT
550 END
560 IFC=2THENCL$="L=" :RETURN
570 CL$="C=" :RETURN
580 PRINTSIN(B(I))*B
590 GOSUB 660
600 PRINT TAB(25)FH$
610 GOTO530
620 PRINTB/SIN(B(I))/4
630 GOSUB 660
640 PRINTTAB(25)FH$
650 GOTO530
660 IFC=2THENFH$="HENRY" :RETURN
670 FH$="FARAD" :RETURN
```

Bild 5. Basic-Programm zum Errechnen der Bauelemente-Werte von Butterworth-Tief- und Hochpaßfiltern

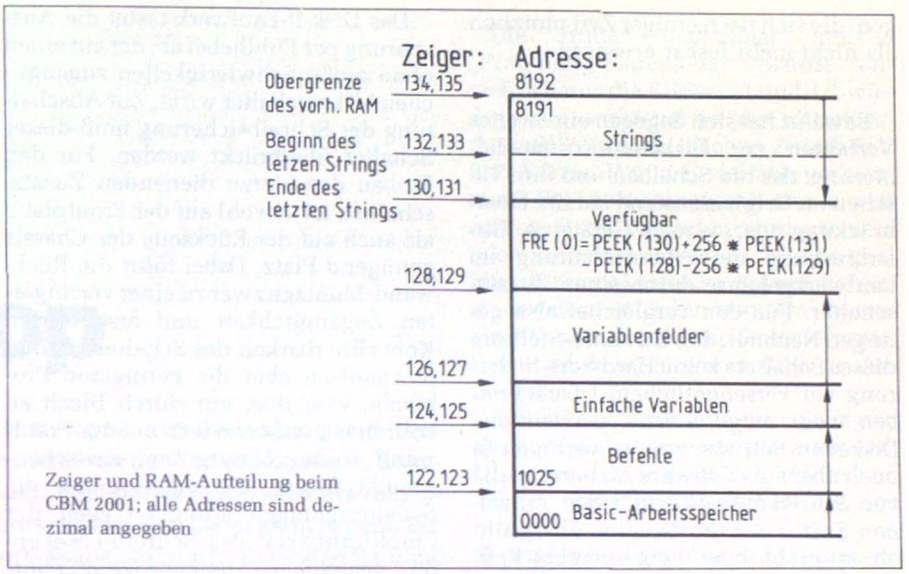
Tips zur Programmierung des PET

Sämtliche Befehle der Computer PET und CBM lassen sich abkürzen, indem man ihren ersten Buchstaben normal und den zweiten zusammen mit der Shift-Taste drückt. Ausnahmen ergeben sich bei PRINT, das mit einem Fragezeichen abgekürzt wird, und PRINT #, was „normal" abgekürzt wird, sowie bei Befehlen, die nicht allein mit zwei Buchstaben unterscheidbar sind. So muß man GOTO als Go und GOSUB als GOs schreiben (wenn man im Grafik-Modus ist, entstehen natürlich Grafiksymbole statt Buchstaben für alle Shift-Zeichen).

Allerdings spart man sich mit solchen Abkürzungen lediglich Tastendrücke bei der Eingabe, nicht aber Speicherplatz: Beim Auflisten erscheinen alle Befehle wieder in voller Länge; sie werden ohnehin nur in Form eines einzigen Byte gespeichert.

In der Neuausgabe der Gebrauchsanweisung des PET wird auf Seite 113 über das programmierte Nachladen eines Programmes referiert. Bei dieser Form des Ladens wird das geladene Programm automatisch gestartet und es wird kein CLR impliziert, d. h., die Variablen gehen nicht verloren, sofern das nachgeladene Programm nicht länger ist als das, in dem der LOAD-Befehl steht. Sollte trotzdem einmal das zweite Programm länger als das erste sein, kann man wie folgt vorgehen:

Man lädt das zweite Programm in den Speicher und fragt mit PEEK die Bytes 124 und 125 ab, in denen der Zeiger auf den Beginn der Variablenliste steht.



Zeiger und RAM-Aufteilung beim CBM 2001; alle Adressen sind dezimal angegeben

Dann wird das erste Programm geladen und an dessen Anfang werden die oben ermittelten Adressen in die Bytes 124 bis 129 mittels POKE eingeschrieben. Folgendes Beispiel soll dies verdeutlichen:

Programm zwei geladen: ?PEEK(124) = 12, ?PEEK (125) = 21.

An den Anfang des ersten Programmes werden nun folgende Zeilen gestellt:

2 POKE124,12 : POKE125,21 :
POKE126,12 : POKE127,21
4 POKE128,12 : POKE129,21

Somit wird sichergestellt, daß alle Variablen des ersten Programmes erst ab der Adresse gespeichert werden, in der keine Befehle des nachzuladenden Programmes mehr eingelesen werden. Die absolute Adresse eines Zeigers läßt sich so ermitteln, hier am Beispiel des Variablen-Beginns gezeigt:

a = PEEK(124) + 256 ·
PEEK(125) = 12 + 256 · 21 = 5388

Die Variablen werden also von Byte 5388 an aufwärts gespeichert.

Jürgen Bonacker

Einbauschalter halbiert Diskettenkosten

Die meisten heute in Gebrauch befindlichen Diskettenstationen für 5-Zoll-Scheiben arbeiten in der Betriebsart „Single-Sided/Single Density", einseitig und mit normaler Informationsdichte beschrieben. Der Gebrauch der Rückseite nach Umstecken der Diskette wird durch den „Write Protect Notch" verhindert, eine ausgestanzte Randkerbe in der Diskettentasche. Bei den meisten Laufwerken tastet ein Fühlhebel die Kerbe ab, andere Geräte prüfen auf optoelektronischem Weg, ob sich am rechten Platz die Aussparung befindet. Vor jeder Schreiboperation vergewissern sich verbreitete Disketten-Betriebssysteme nun vom Vorhandensein der Kerbe und verweigern gegebenenfalls das Schreiben, stattdessen ergehen Fehlermeldungen wie „Write Protected".

Findige Tüftler mit Sinn für Sparsamkeit fanden heraus, daß man mit Hilfe eines spitzen Messers auch auf der dem „Notch" gegenüberliegenden Seite der Diskettentasche eine solche Markierung aus dem Plastikmaterial herausschneiden und so dann auch die zweite Diskettentasche benutzen kann. Diese Operation verlangt jedoch einiges Geschick, weil bei unpräzisem Arbeiten allzuleicht die Magnetscheibe selbst beschädigt wird oder Materialpartikel in das Taschenninnere gelangen – mit bösen Folgen für, wie die Erfahrung lehrt, gerade wertvolle Aufzeichnun-

Baut schnellere Rechner!

Die Mikrocomputer-Technk schreitet fort und tendiert in letzter Zeit immer mehr zu größerer Datenbreite: 4-bit-Prozessoren gelten (manchmal unbegründet) als veraltet, 8-bit-Prozessoren leiden zum Teil schon sehr unter 16-bit-Konkurrenz, und selbst 32-bit-CPUs sind im Gespräch, CPUs also, die 32 bit parallel verarbeiten können – eine enorme Geschwindigkeitszunahme ist die Folge.

Demgegenüber sind selbst modernste Taschenrechner Steinzeit-Geräte. Denn sie arbeiten üblicherweise bitseriell: Ein Bit nach dem anderen, sozusagen 1-bit-Prozessoren also, die dementsprechend langsam sind. Natürlich hat das gute Gründe, denn in kleinen Rechnern haben nur kleine ICs Platz, und je größer die zu verarbeitende Datenbreite ist, desto mehr Pins müssen die Rechner-ICs haben.

Trotzdem muß man sich fragen, ob es mit der Vorstellung von technischem Fortschritt zu vereinen ist, daß gerade programmierbare Rechner eben wegen dieser bitweisen Arbeitsweise ungeheuer langsam sind und bereits für die Aufgabe, ganz einfach nichts zu tun (NOP-Befehl), etliche Millisekunden benötigen. Wie schön wäre es, wenn für eine einfache Integration mit sechsstelliger Genauigkeit nicht die gesamte Batteriekapazität des programmierbaren Rechners aufgewendet werden müßte...

Fe.

gen, die sich nach einiger Zeit plötzlich als nicht mehr lesbar erweisen.

Bewährt hat sich dagegen ein zweites Verfahren zur Disketten-Kosten-Halbierung, das die Scheiben und ihre Taschen im Originalzustand beläßt: Überbrückung oder, je nach Gerätetyp, Unterbrechung der Abtastschaltung im Laufwerkgehäuse durch einen Zusatzschalter. Für den vergleichsweise geringen Nachteil, daß bei Aktiv-Stellung dieses Schalters keine Hardware-Sicherung vor versehentlichem Überschreiben mehr möglich ist – neuzeitliche Disketten-Betriebssysteme verfügen ja noch über eine Software-Sicherung, die vor Schreiboperationen jeden einzelnen Sektor auf der Scheibe überprüft, ob er nicht eine Schreibverbot-Kennung trägt – handelt man sich also einen handfesten finanziellen Vorteil ein.

Erprobt wurde das Verfahren an mehreren Laufwerken „Disk II" mit Maschinen ITT 2020 und Apple II und den Disketten-Betriebssystemen DOS 3.1, DOS 3.2 und DOS 3.21; die doppelt genutzten Scheiben zeigten in allen Fällen die gleiche Betriebszuverlässigkeit wie ihre einseitigen Geschwister.

Das Disk-II-Laufwerk tastet die Aussparung per Fühlhebel ab, der auf einen ohne große Schwierigkeiten zugänglichen Mikroschalter wirkt. Zur Abschaltung der Schreibsicherung muß dieser Schalter überbrückt werden. Für den Einbau des hierzu dienenden Zusatzschalters ist sowohl auf der Frontplatte als auch auf der Rückseite des Chassis genügend Platz. Dabei führt die Rückwand-Montage zwar zu einer verringerten Zugänglichkeit und erschwerten Kontrollierbarkeit des Schalters, bringt mechanisch aber die geringeren Probleme, weil dort nur durch Blech zu bohren ist, während sich auf der Frontwand eine Kunstgummi-Zwischenschicht findet. Wo auch immer die Schaltermontage erfolgt: Wegen der Empfindlichkeit des Schreib-Lesekopfes gegenüber Staubpartikeln bohrt man am besten im Luftstrom eines Staubsaugers (hierbei bitte den Unterdruck-Anschluß benutzen, nicht den „Blas-Anschluß"). Die Verbindungsleitungen Mikroschalter-Zusatzschalter sind elektrisch unkritisch und weder gegenüber dem im Gehäuse herrschenden elektromagnetischen Störnebel empfindlich, noch verursachen sie selbst Störungen auf anderen Leitungen.

Hans-Georg Joepgen

Verschiebe-Lader für den AIM-65

Bei der Programmierung von EPROMs und ähnlichen Aufgaben muß sich das Objekt-Programm im RAM befinden. Bei der Assemblierung eines solchen Programms ergeben sich Schwierigkeiten, wenn der Adreßbereich des für das EPROM bestimmten Programms nicht im RAM-Bereich liegt oder sogar mit dem ROM-Bereich kollidiert. Hier hilft nur das Assemblieren direkt auf Band.

Die vorliegende Routine ermöglicht es nun, ein so generiertes Objektprogramm im AIM-Format vom Tonband in einen beliebigen RAM-Bereich zu laden, ohne darin evtl. enthaltene absolute Sprungadressen zu ändern. Die Routine – die selbst übrigens voll verschieblich ist – arbeitet wie folgt: Die im ersten Datenblock erscheinende Adresse wird mit der vom Anwender gewünschten Startadresse verglichen und daraus der erforderliche Offset errechnet. Jeder Block des vom Band eingelesenen Programms wird nun entsprechend umgerechnet und in den neuen Adreßbereich eingelesen.

Da der Offset nur einmal am Anfang errechnet wird, muß vorausgesetzt werden, daß das Objektprogramm nicht aus mehreren Segmenten mit weit auseinanderliegenden Adreßteilen besteht. Eine Ausnahme berücksichtigt die Routine allerdings: Wenn sich im ersten Block Adressen im Bereich der PAGE 1 befinden, also die Werte der Funktionstasten bzw. Input- und Output-Handler, so werden diese nicht verändert, sondern normal eingelesen. Dadurch wird auch das Einlesen vorhandener Objektprogramme mit einem solchen Vorspann ermöglicht. Eventuell vorhandene absoluten Sprungbefehle müssen dann noch manuell oder mit einem der bekannten Umrechnungsprogramme geändert werden.

Vor dem Start muß zuerst der User-Input-Handler in 0108/0109 auf die Startadresse der Routine, hier 0200, gerichtet werden. Nach dem Aufruf der Laderoutine wird der Prompt „<L>IN=" mit „U" statt „T" beantwortet. Der AIM reagiert mit dem Prompt „TO=" für die Eingabe der gewünschten Startadresse. Alle weiteren Angaben erfolgen wie üblich.

Es werden drei Plätze in der Zero-Page benutzt, nämlich 50/51 für die neue Startadresse bzw. den errechneten Offset, und 52 als Offset-Flag. Noch ein Hinweis am Rande: Ein Assemblieren auf Band mit dem KIM-Format ist – entgegen der Angabe im Handbuch – nicht möglich.

H. Steder

```
=<M>=0200  B0 1A 20 3E E8 20 A7 E7 AD 1C A4 85 50 AD 1D A4
=< > 0210  85 51 A2 00 86 52 20 A2 E8 4C 2F E3 CE 12 A4 20
=< > 0220  3B ED C9 3B D0 F9 20 4D EB 20 4B E5 AA 20 4B E5
=< > 0230  8D 1D A4 20 4B E5 8D 1C A4 AD 1D A4 C9 01 F0 26
=< > 0240  A5 52 D0 11 38 AD 1C A4 E5 50 85 50 AD 1D A4 E5
=< > 0250  51 85 51 E6 52 38 AD 1C A4 E5 50 8D 1C A4 AD 1D
=< > 0260  A4 E5 51 8D 1D A4 18 8A D0 06 20 21 E3 4C 82 E1
=< > 0270  20 FD E3 20 13 E4 CA D0 F7 20 FD E3 CD 1F A4 F0
=< > 0280  03 4C 85 E3 20 FD E3 CD 1E A4 F0 93 4C 85 E3 00
```

Was für KIM-Besitzer kein Problem war, macht beim AIM-65 dieses Maschinenprogramm erforderlich – das Laden eines Programms von der Kassette mit veränderten Adressenbereichen

µC-Nachrichten

Die Braunschweiger Firma Computer Studio bietet als erster Lieferant in Deutschland eine hochauflösende Grafik für den Mikrocomputer TRS-80 von Tandy an. Die Platine mit der hierzu erforderlichen Hardware kostet allerdings rund 700 DM.

F'

Ein Buchhaltungsprogramm mit dem Namen FIBU-80 vertreibt die Holzkirchener Software-Firma Redysoft für die Mikrocomputer TRS-80 Modell I und II. Es arbeitet mit einem frei bestimmbaren Kontenrahmen. Damit ist es u. a. möglich, einen kompletten betriebswirtschaftlichen Jahresabschluß durchzuführen. Anfang Dezember will Redysoft auch ein Fakturier- und Lagerhaltungsprogramm vorstellen.

F'

Drei neue Computer gibt es bei Tandy/Radio Shack: Einen Basic-Taschencomputer, bei dem es sich um den Sharp-Rechner PC-1210 handelt, der u. a. auch von Quelle vertrieben wird; einen recht preiswerten Computer mit Ausgang für Farbfernsehgeräte („Color Computer", in den USA ca. 400 Dollar); und das für professionelle Anwender gedachte Modell III (US-Preis 2500 Dollar) mit 32 K und 2 Disks.

Reinhard Keller

Kugelkopf-Schreibmaschine als Drucker

Der folgende Beitrag beschreibt den Anschluß einer IBM-Kugelkopf-Schreibmaschine an einen beliebigen 6502-Computer, z. B. KIM-1 oder AIM-65; auch an den PET-2001 oder an den CBM-3001 läßt sich die Software adaptieren.

Früher oder später wird jeder Benutzer eines Mikrocomputers sich wünschen, seine Programme oder Texte drucken zu lassen. Hier bestehen drei Möglichkeiten:

1. Verwendung eines Fernschreibers, was die preisgünstigste Lösung darstellen würde. Leider ist er sehr laut und kann entweder nur große oder nur kleine Buchstaben drucken, was zum Schreiben von Briefen usw. ungünstig ist.
2. Der Kauf eines schnellen und leisen Druckers, der auch ein gutes Schriftbild bietet, wird wohl für die meisten Mikrocomputerbenutzer eine zu große finanzielle Aufwendung sein. Ferner ist die Druckqualität z. B. für Briefe noch nicht ausreichend.
3. Hier bieten sich gebrauchte IBM-Kugelkopfschreibmaschinen (Typen 072 I/O, 073 I/O) an, die von verschiedenen Second-Hand-Geschäften zwischen 600 und 1000 DM angeboten werden. Diese Maschinen sind verhältnismäßig leise und haben eine sehr gute Schriftqualität.

Der Anschluß einer solchen Input/Output-IBM-Schreibmaschine an einen AIM oder KIM soll hier beschrieben werden.

Arbeitsweise einer IBM-I/O-Schreibmaschine

Alle Funktionen, die bei einer Schreibmaschine normalerweise mit Hilfe der Tasten rein mechanisch oder bei einer „elektrischen Schreibmaschine" motorisch unterstützt ausgeführt werden, können bei Input/Output-Maschinen auch durch Elektromagnete betätigt werden.

Diese Elektromagnete, die mit 48 V betrieben werden, werden über Reed-Relais von den Ports des Mikrocomputers angesteuert.

Für die Kontrollfunktionen einer Schreibmaschine, wie Leertaste, Wagenrücklauf, Zeilenschaltung, Rücktaste, Tabulator, Umschaltung auf Großbuchstaben und Kleinbuchstaben, ist in der Maschine je ein Elektromagnet vorhanden. Wenn also der entsprechende Port des Computers angesprochen wird, schaltet ein Reed-Relais die Betriebsspannung für einen Elektromagnet ein, und die gewünschte Funktion wird von der Schreibmaschine ausgeführt.

Für das Drucken der Buchstaben befinden sich in der Maschine ebenfalls 7 Magnete (T1, T2, R1, R2, R2A, R5, CK), die den Kugelkopf drehen und neigen, bis der gewünschte Buchstabe dem Schreibpapier zugewandt ist. Nun wird der Magnet aktiviert, der den Anschlag ausführt. Die Anschlüsse für alle Magnete befinden sich auf der Rückseite der Schreibmaschine an einem 50-Pin-Stecker.

Für die IBM-Schreibmaschine sind viele verschiedene Typenköpfe erhältlich. Hier wird ein COURIER 72 verwendet, auf den sich auch die im Drukkerprogramm verwendeten Bitmuster beziehen. Auf den Typenkugeln sind alle Großbuchstaben auf der einen und alle Kleinbuchstaben auf der anderen Kugelhälfte angeordnet. In *Bild 1* ist eine „Abwicklung" des COURIER-Kopfes zu sehen.

Das Interface

Das Interface (*Bild 2*) ist eine einfache Steuerlogik mit 14 DIL-Reedrelais (Typ SRW.5). Anstelle dieser Reedrelais können auch Optokoppler verwendet werden. Sie haben die Aufgabe, mit den TTL-Pegeln der Gatter die 48 V (ca. 150 mA) für die Magnete der IBM-I/O-Maschine zu schalten. Die Stromversorgung für die Magnete muß 48 V Gleichstrom liefern, ist unstabilisiert und sie muß, für den Fall daß 7 Magnete angesprochen werden, ca. 1 A abgeben können.

Außerdem sind viele IBM-Schreibmaschinen für 110 V Netzspannung vorgesehen. In diesen Fällen wird noch ein Trafo 220/110 V erforderlich.

Aus Bild 2 geht hervor, daß das Bit 0 des Ausgabeports zum Umschalten auf 2 Gruppen von Reedrelais benutzt wird. Wenn das Bit 0 low ist, so ist die linke Relaisgruppe für die Kontrollfunktionen betriebsbereit; ist das Bit 0 high, ist die Gruppe der Relais schaltbereit, die

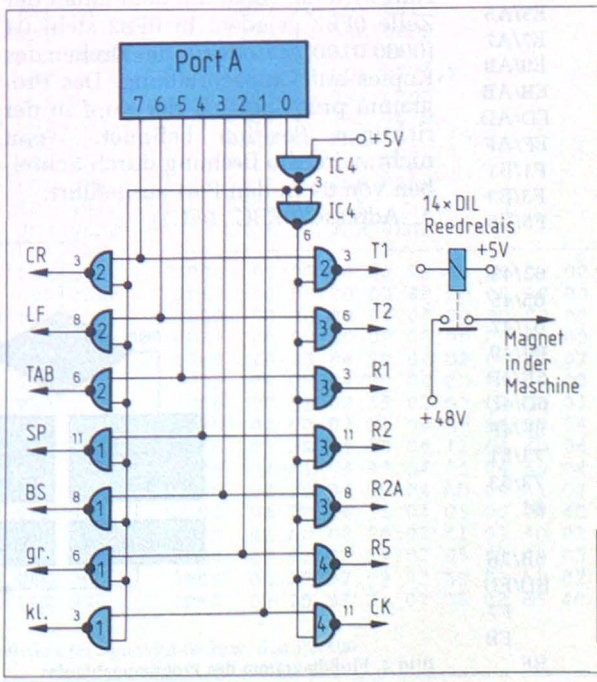


Bild 2. Externe Logik zur Ansteuerung der Kugelkopf-Schreibmaschine

		Bit					
7	6	5	4	3	2	1	0
T1	T2	R1	R2	R2A	R5	CK	1
CR	LF	TAB	SP	BS	gr.	kl.	0

V	/	C	&	"	Z	§	?	)	*	£
B	H	K	E	N	T	L	C	D	U	X
W	S	I	:	=	§	O	A	R	V	M
_	Y	Q	P	+	J		^	F	G	
9	0	6	5	2	z	4	8	7	3	$
b	h	k	e	n	t	l	c	d	u	x
w	s	i	ä	.	ü	o	a	r	v	m
-	y	q	p	ö	j	,		f	g	

Bild 1. Zeichenvorrat des IBM-Kugelkopfes Courier 72

für das Drucken der Buchstaben zuständig ist. *Bild 3* zeigt den Anschluß des 50-Pin-Steckers.

Soll die Schreibmaschine z. B. einen Wagenrücklauf ausführen, muß Port 0 auf low und Port 7 auf high gehen. In den Ausgangsport muß also das Bitmuster 1000 0000 geschrieben werden. Sind aber Port 0 und Port 7 high (Bitmuster 1000 0001), wird der Magnet T1 aktiv und ein W gedruckt.

Für das Drucken von Buchstaben ist in den meisten Fällen eine Kombination von mehreren Magneten erforderlich. Soll beispielsweise ein Z entstehen, müssen die Magnete T1, T2, R1, R2, R2A aktiviert werden und Port 0 high sein. Jeder Buchstabe hat somit ein bestimmtes Bitmuster (Z = 1111 1001). Die Bitmuster für die Buchstaben und Ziffern sind in der *Tabelle* zusammengestellt.

Die Umlaute „ä" erreicht man durch SHIFT 7; „ö" durch SHIFT ↑ und „ü" durch SHIFT (.

Tabelle: Codierung der Zeichen

		SHIFT		
CHAR.	CODE	CHAR.	CODE	Speicherzellen
a	8D	A	8D	0FC3–/83
b	41	B	41	C5/85
c	4D	C	4D	C7/87
d	6D	D	6D	C9/89
e	69	E	69	CB/8B
f	1D	F	1D	CD/8D
g	3D	G	3D	CF/8F
h	61	H	61	D1/91
i	89	I	89	D3/93
j	05	J	05	D5/95
k	49	K	49	D7/97
l	65	L	65	D9/99
m	BD	M	BD	DB/9B
n	59	N	59	DD/9D
o	A5	O	A5	DF/9F
p	29	P	29	E1/A1
q	09	Q	09	E3/A3
r	AD	R	AD	E5/A5
s	A1	S	A1	E7/A7
t	45	T	45	E9/A9
u	5D	U	5D	EB/AB
v	9D	V	9D	ED/AD
w	81	W	81	EF/AF
x	7D	X	7D	F1/B1
y	21	Y	21	F3/B3
z	C5	Z	C5	F5/B5
1	65	£	FD	63/43
2	D9	"	D9	65/45
3	DD	§	85	67/47
4	E5	$	FD	69/49
5	E9	%	E5	6B/4B
6	C9	&	E9	6D/4D
7	ED	ä	A9	6F/4F
8	CD	(	C9	71/51
9	C1	)	ED	73/53
0	E1			61
–	03	=	99	5B/7B
^	2D	ö	19	BD/FD
		ü	85	F7
		1/	C1	FB
–	03			BF

Das Druckerprogramm

Die Aufgabe des Druckerprogramms ist es nun, ein im Akku befindliches ASCII-Zeichen oder eine Kontrollfunktion in einen der IBM-Maschine entsprechenden Code umzuwandeln, diesen Code in den Ausgabeport zu senden und damit die Magnete in der Maschine zu aktivieren, und eine Verzögerung hervorzurufen, die für die Ausführung der Arbeitsvorgänge in der Schreibmaschine erforderlich ist.

Diese Verzögerung ist für die Kontrollfunktionen und Druckvorgänge sehr unterschiedlich. Darum muß das Programm prüfen, welcher Vorgang abläuft und eine entsprechend lange Verzögerung einleiten. Das Flußdiagramm (*Bild 4*) gibt einen Überblick vom Ablauf des Druckerprogramms.

Das vorliegende Programm benutzt den Port A des Computers für die Steuerung der I/O-Schreibmaschine. Alle vorkommenden Codes für den COURIER-Typenkopf sind am Ende des Programms zusammengefaßt, und liegen im Speicherbereich von 0F00...0FFF.

Der Zugriff des Druckerprogramms auf die Codetabelle am Beispiel „A":

1. Adresse 0E6B (ASL)
Der Akku-Inhalt wird verdoppelt, z. B. „A"=41 = Bitmuster 0100 0001 wird zu 1000 0010 = 82.
2. Adresse 0E6C (TAY)
Akku-Inhalt wird in das Y-Register transferiert.
3. Adresse 0E6D (LDA 0F00,Y)
Nun wird der Akku mit dem Inhalt der Zelle 0F82 geladen. In 0F82 steht 04 (0000 0100) = Code für das Drehen des Kopfes auf Großschreibung. Das Programm prüft, ob sich der Kopf in der richtigen Position befindet. Wenn nicht, wird die Drehung durch Schreiben von 04 in den Port ausgeführt.
4. Adresse (0EBC...0EC1)

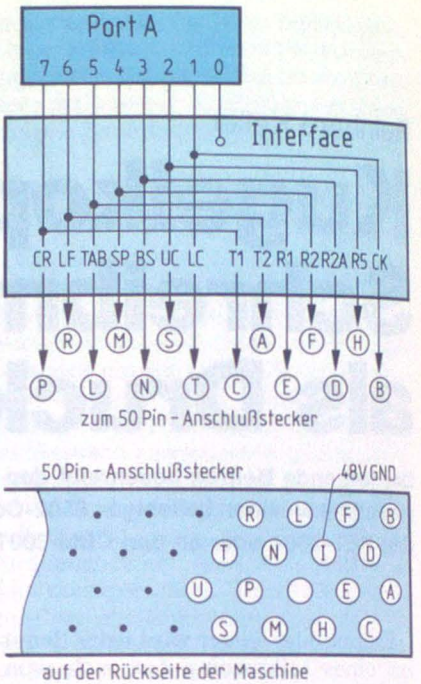


Bild 3. Das Interface wird über einen 50poligen Stecker mit der Schreibmaschine verbunden

Danach wird der bereits um 1 erhöhte (0E70) Inhalt des Y-Registers aus der Hilfszelle 0096 wieder geladen und ist nun 83. Mit LDA 0F00,Y (0EBE) ladet der Akku nun aus der Codetabelle (0F83) den Wert 8D (1000 1101). Dieses Bitmuster aktiviert die Magnete T1, R2A und R5, was zum Drucken des „A" führt.

Die Programmzeilen von 0E00...0E61 bieten als Anregung drei von vielen Möglichkeiten zur Anwendung:

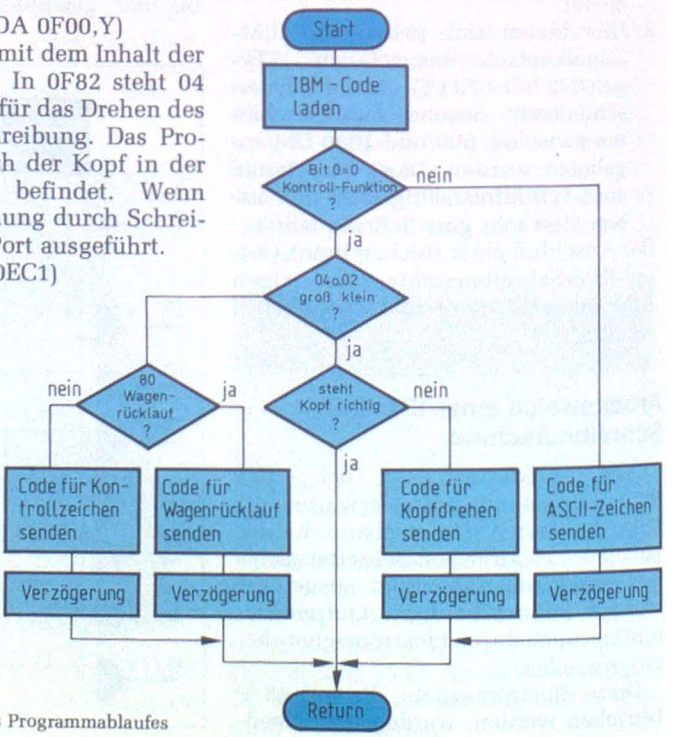


Bild 4. Flußdiagramm des Programmablaufes

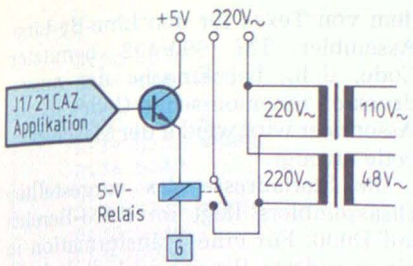


Bild 5. Beim AIM-65 läßt sich über die VIA-Leitung CA 2, gesteuert vom PCR-Register (A00C), die Stromversorgung der Schreibmaschine und des Interface ein- und ausschalten

1. Mit der Taste F1 (010C... = JMP 0E25) wird die AIM-65-Terminal- oder Computertastatur eingeschaltet.
2. Mit der Taste F2 (010F... = JMP 0E34) arbeitet die IBM-Schreibmaschine als Hardcopy-Terminal. Ein zweites betätigen der F2-Taste beendet diesen Modus.
3. Mit der Taste F3 (0112... = JMP 0E09) kann die Schreibmaschine ein- und ausgeschaltet werden, sofern die Schaltung nach *Bild 5* realisiert ist.

Abschließend noch einige Besonderheiten von Wichtigkeit. Wenn man es nicht vorzieht, die Schreibmaschine nach Bild 5 durch das Programm ein- und ausschalten zu lassen und besonders auch für die ersten Versuche, sollte erst die F1-Taste betätigt werden, ehe die Netzspannung für die Maschine angeschlossen wird, um ein unkontrolliertes Loslaufen zu vermeiden.

Außerdem darf die 48-V-Spannung für die Magnete nicht eingeschaltet sein, wenn die Schreibmaschine ausgeschaltet ist. Das hat folgenden Grund: Die Magnete bleiben bei stehender Maschine in einem undefinierten Zustand und ziehen dann einen Dauerstrom, der den 48-V-Trafo überlastet.

Die Funktion als Hardcopy-Terminal arbeitet nur, wenn die AIM-Tastatur aktiv ist, das heißt, wenn der Schalter TTY-KB auf KB geschaltet ist.

Das Programmlisting in *Bild 6* entstand auf einem AIM-65 mit angeschlossener Kugelkopf-Schreibmaschine. Will man es für den KIM-1 umschreiben, so muß man die I/O-Port-Adressen anpassen. Beim AIM-65 sind dies: A001 = Port-A-Datenregister, A003 = Port-A-Datenrichtungsregister. An Systemroutinen des AIM wurden verwendet: E93C = Einlesen eines ASCII-Zeichens von der Tastatur, E97A = Ausgabe eines ASCII-Zeichens auf das Display, EEA8 = Ausgabe eines ASCII-Zeichens an ein ev. angeschlossenes serielles ASCII-Terminal. A406/A407 ist ein RAM-Vektor, über den das AIM-Monitorprogramm normalerweise zur Routine für die Display-Ausgabe springt und der hier für die Schreibmaschinen-Ausgabe entsprechend umgesetzt wird. Die Adressen A00C, A00D, A009 gehören zum VIA-Timer 1 (User-6522). FEE9 ist die Einsprungadresse des AIM-Monitorprogramms.

Beim Umschreiben des Programms für den KIM-1 ist die Verzögerungsroutine zu ändern, da der KIM nicht über ein 6522-VIA verfügt. Für PET und CBM ergeben sich nur geringfügige Änderungen, wobei allerdings ein Ausdruck parallel zur Display-Ausgabe schwierig ist, weil hier kein RAM-Vektor zur Display-Routine existiert. In der vorliegenden Form verwendet das Programm die Zero-Page-Adressen 0096 und 0097. Die Sprünge bei 010C sind den Tasten F1, F2 und F3 zugeordnet.

```
K)*=010C
/03
 010C 4C JMP 0E25
 010F 4C JMP 0E34
 0112 4C JMP 0E09

K)*=0E00
/
 0E00 AC LDY A00C
 0E03 C0 CPY §0C
 0E05 F0 BEQ 0E1A
 0E07 D0 BNE 0E0F
 0E09 20 JSR 0E00
 0E0C 4C JMP FEE9
 0E0F A2 LDX §FF
 0E11 8E STX A003
 0E14 A2 LDX §0C
 0E16 8E STX A00C
 0E19 60 RTS
 0E1A A2 LDX §30
 0E1C 20 JSR 0EEB
 0E1F A2 LDX §00
 0E21 8E STX A00C
 0E24 60 RTS
 0E25 20 JSR 0E0F
 0E28 20 JSR E93C
 0E2B 20 JSR E97A
 0E2E 20 JSR 0E62
 0E31 4C JMP 0E25
 0E34 AD LDA A406
 0E37 C9 CMP §05
 0E39 F0 BEQ 0E4B
 0E3B A9 LDA §05
 0E3D 8D STA A406
 0E40 A9 LDA §EF
 0E42 8D STA A407
 0E45 20 JSR 0E1F
 0E48 4C JMP FEE9
 0E4B 20 JSR 0E0F
 0E4E A9 LDA §5B
 0E50 8D STA A406
 0E53 A9 LDA §0E
 0E55 8D STA A407
 0E58 4C JMP FEE9
 0E5B 20 JSR EEA8
 0E5E 20 JSR 0E62
 0E61 60 RTS
 0E62 48 PHA
 0E63 85 STA 96
 0E65 8A TXA
 0E66 48 PHA
 0E67 98 TYA
 0E68 48 PHA
 0E69 A5 LDA 96
 0E6B 0A ASL .A
 0E6C A8 TAY
 0E6D B9 LDA 0F00,Y
 0E70 C8 INY
 0E71 84 STY 96
 0E73 4A LSR .A
 0E74 90 BCC 0E8B
 0E76 2A ROL .A
 0E77 A2 LDX §FF
 0E79 A0 LDY §01
 0E7B 20 JSR 0EDF
 0E7E A2 LDX §50
 0E80 A0 LDY §01
 0E82 20 JSR 0EEB
 0E85 68 PLA
 0E86 A8 TAY
 0E87 68 PLA
 0E88 AA TAX
 0E89 68 PLA
 0E8A 60 RTS
 0E8B 2A ROL .A
 0E8C C9 CMP §02
 0E8E F0 BEQ 0EA9
 0E90 C9 CMP §04
 0E92 F0 BEQ 0EA9
 0E94 C9 CMP §80
 0E96 F0 BEQ 0EC7
 0E98 A2 LDX §30
 0E9A A0 LDY §01
 0E9C 20 JSR 0EDF
 0E9F A2 LDX §F0
 0EA1 A0 LDY §01
 0EA3 20 JSR 0EEB
 0EA6 4C JMP 0E7E
 0EA9 C5 CMP 97
 0EAB F0 BEQ 0EB5
 0EAD 85 STA 97
 0EAF A2 LDX §A0
 0EB1 A0 LDY §02
 0EB3 D0 BNE 0EB9
 0EB5 A2 LDX §80
 0EB7 A0 LDY §01
 0EB9 20 JSR 0EDF
 0EBC A4 LDY 96
 0EBE B9 LDA 0F00,Y
 0EC1 20 JSR 0EDB
 0EC4 4C JMP 0E7E
 0EC7 48 PHA
 0EC8 A9 LDA §0A
 0ECA 20 JSR EEA8
 0ECD 68 PLA
 0ECE 20 JSR 0EDB
 0ED1 A2 LDX §C0
 0ED3 A0 LDY §0A
 0ED5 20 JSR 0EEB
 0ED8 4C JMP 0E7E
 0EDB A2 LDX §80
 0EDD A0 LDY §01
 0EDF 8D STA A001
 0EE2 20 JSR 0EEB
 0EE5 A9 LDA §00
 0EE7 8D STA A001
 0EEA 60 RTS
 0EEB 8A TXA
 0EEC 48 PHA
 0EED AA TAX
 0EEE 8E STX A009
 0EF1 A9 LDA §20
 0EF3 2C BIT A00D
 0EF6 F0 BEQ 0EF3
 0EF8 68 PLA
 0EF9 48 PHA
 0EFA 88 DEY
 0EFB D0 BNE 0EED
 0EFD 68 PLA
 0EFE 60 RTS

0F00  00 00 00 02 02 00 00 02 00 00 00 00 00 00 00 00
0F10  00 00 00 00 40 00 00 00 00 00 80 00 00 00 00 00
0F20  00 00 00 00 00 00 00 00 00 00 00 00 0F 00 00 00
0F30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0F40  10 33 04 FD 04 D9 04 85 02 FD 04 E5 04 E9 02 A9
0F50  04 C9 04 ED 04 DD 04 19 02 15 02 03 02 99 04 E1
0F60  02 E1 02 65 02 D9 02 DD 02 E5 02 E9 02 C9 02 ED
0F70  02 CD 02 C1 04 A9 04 ED 04 C9 04 99 04 ED 04 CD
0F80  F3 3B 04 8D 04 41 04 4D 04 6D 04 69 04 1D 04 3D
0F90  04 61 04 89 04 05 04 49 04 65 04 BD 04 59 04 A5
0FA0  04 29 04 09 04 AD 04 A1 04 45 04 5D 04 9D 04 81
0FB0  04 7D 04 21 04 C5 00 00 4C 5F 04 03 04 2D 04 03
0FC0  B1 A0 02 8D 02 41 02 4D 02 6D 02 69 02 1D 02 3D
0FD0  02 61 02 89 02 05 02 49 02 65 02 BD 02 59 02 A5
0FE0  02 29 02 09 02 AD 02 A1 02 45 02 5D 02 9D 02 81
0FF0  02 7D 02 21 02 C5 02 85 4C 4D 04 C1 02 19 4D 4F
```

Bild 6. Programmlisting der nötigen Steuerungssoftware für den AIM-65 bzw. den PC-100

Hans-Dieter Boels, Achim Schafmeister

Disassembler für TM-990-Mikrocomputer

Jeder Anwender von Mikrocomputern wird aus eigener Erfahrung bestätigen können, daß bei der Erstellung längerer Programme ein exaktes Programmprotokoll unerläßlich ist. Da nun leider die meisten Programme erst nach einigen Korrekturen wunschgemäß laufen, muß das Programmprotokoll ständig per Hand aktualisiert werden. Deshalb haben die Verfasser einige Stunden Zeit investiert und als Alternative zu der von Texas Instruments auf der Entwicklungsplatine 990/302 angebotenen Disassembler-Software einen eigenen Disassembler programmiert.

Der Disassembler belegt einen Speicherplatz von 1 KByte und kann je nach Verfügbarkeit fest in EPROMs programmiert oder bei Gebrauch von üblichen Peripheriespeichern in den RAM-Bereich eingelesen werden. Der vom Disassembler ausgedruckte mnemonische Code ist voll kompatibel zu dem von Texas für den Line-By-Line-Assembler TM 990/402 benutzten Code, d. h., bei Eingabe des ausgedruckten mnemonischen Codes in den Assembler wird wieder der Maschinencode erzeugt.

Die Startadresse des vorgestellten Disassemblers liegt im RAM-Bereich auf D000. Für eine Transformation in einen anderen Bereich ist lediglich die Abänderung der Branch- und XOP-Befehle notwendig. Der Disassembler ist nach Einlesen des Programms startbereit, wenn der Programmcounter auf D000 gesetzt wird. Der Beginn des Arbeitsbereiches wird am Anfang der Programmausführung selbständig auf WP = FE00 gesetzt. Steht dieser Bereich nicht zur Verfügung, ist das Programm entsprechend zu verändern. Wie üblich wird es dann mit E gestartet. Der Mikrocomputer meldet sich mit:

E VON

Dann wartet er auf die Eingabe der Anfangsadresse des zu disassemblierenden Programms. Nach Eingabe dieser Adresse, z. B. C000, erfolgt der Ausdruck:

E VON C000 BIS

Jetzt erwartet der Computer die Adresse des letzten zu disassemblierenden Be-

```
E VON D000 BIS D072  $/+
  D000 02E0 LWPI >FE00
  D002 FE00
  D004 04C2 CLR R2
  D006 0200 LI R0,>0006
  D008 0006
  D00A 02E0 LWPI >FF00
  D00C FF00
  D00E 0205 LI R5,>FE00
  D010 FE00
  D012 0206 LI R6,>D1F8
  D014 D1F8
  D016 2FA0 XOP @>D086,R14
  D018 D086
  D01A 2E41 XOP R1,R9
  D01C D020
  D01E D016
  D020 2FA0 XOP @>D08C,R14
  D022 D08C
  D024 2E42 XOP R2,R9
  D026 D02A
  D028 D020
  D02A 2FA0 XOP @>D1EE,R14
  D02C D1EE
  D02E 2EC4 XOP R4,R11
  D030 0284 CI R4,>2400
  D032 2400
  D034 1602 JNE $>D03A
  D036 0206 LI R6,>D060
  D038 D060
  D03A 0284 CI R4,>2B00
  D03C 2B00
  D03E 1602 JNE $>D044
  D040 0206 LI R6,>D2FC
  D042 D2FC
  D044 2FA0 XOP @>D092,R14
  D046 D092
  D048 2E81 XOP R1,R10
  D04A 2F20 XOP @>D1CC,R12
  D04C D1CC
  D04E 2E91 XOP *R1,R10
  D050 2F20 XOP @>D1CC,R12
  D052 D1CC
  D054 0405 BLWP R5
  D056 05C1 INCT R1
  D058 8081 C R1,R2
  D05A 12F4 JLE $>D044
  D05C 0460 B @>0080
  D05E 0080
  D060 C060 MOV @>FF02,R1
  D062 FF02
  D064 8081 C R1,R2
  D066 1B01 JH $>D06A
  D068 0380 RTWP
  D06A C081 MOV R1,R2
  D06C 2FA0 XOP @>D09C,R14
  D06E D09C
  D070 C120 MOV @>FF04,R4
  D072 FF04
?
 E VON D074 BIS D084  $/+
  D074 D0F2 MOVB *R2+,R3
  D076 1303 JEQ $>D07E
  D078 2F03 XOP R3,R12
  D07A 8102 C R2,R4
  D07C 12FB JLE $>D074
  D07E D0D2 MOVB *R2,R3
  D080 1301 JEQ $>D084
  D082 0602 DEC R2
  D084 0380 RTWP
?E VON D086 BIS D0DF  $/+ $
  D086 564F $VON
  D088 4E20
  D08A 0000
  D08C 4249 $BIS
  D08E 5320
  D090 0000
  D092 0D0A $
?
  D094 2020
  D096 2000
  D098 4C49 $LI
  D09A 0020
  D09C 2400 $$
  D09E 4149 $AI
  D0A0 0000
  D0A2 3100 $1
  D0A4 414E $ANDI
  D0A6 4449
  D0A8 0000
  D0AA 4F52 $ORI
  D0AC 4900
  D0AE 2000 $
  D0B0 4349 $CI
  D0B2 0000
  D0B4 0000 $
  D0B6 5354 $STWP
  D0B8 5750
  D0BA 0000
  D0BC 5354 $STST
  D0BE 5354
  D0C0 0000
  D0C2 4C57 $LWPI
  D0C4 5049
  D0C6 0000
  D0C8 4C49 $LIMI
  D0CA 4D49
  D0CC 0000
  D0CE 4944 $IDLE
  D0D0 4C45
  D0D2 0000
  D0D4 5253 $RSET
  D0D6 4554
  D0D8 0000
  D0DA 5254 $RTWP
  D0DC 5750
  D0DE 0000
E VON D0E0 BIS D151  $/+ $
  D0E0 434B $CKON
  D0E2 4F4E
  D0E4 0000
  D0E6 434B $CKOF
  D0E8 4F46
  D0EA 0000
  D0EC 4C52 $LREX
  D0EE 4558
  D0F0 0000
  D0F2 424C $BLWP
  D0F4 5750
  D0F6 0000
  D0F8 4200 $B
  D0FA 002B
  D0FC 3E00 $>
  D0FE 5800 $X
  D100 203E $ >
  D102 0000
  D104 434C $CLR
  D106 5200
  D108 0000
  D10A 4E45 $NEG
  D10C 4700
  D10E 0000
  D110 494E $INV
  D112 5600
  D114 0000
  D116 494E $INC
  D118 4300
  D11A 0000
  D11C 494E $INCT
  D11E 4354
  D120 0000
  D122 4445 $DEC
  D124 4300
  D126 0000
  D128 4445 $DECT
  D12A 4354
```

```
D12C 0000
D12E 424C $BL
D130 0000
D132 0000 $
D134 5357 $SWPB
D136 5042
D138 0000
D13A 5345 $SETO
D13C 544F
D13E 0000
D140 4142 $ABS
D142 5300
D144 0000
D146 5352 $SRA
D148 4100
D14A 5352 $SRL
D14C 4C00
D14E 534C $SLA
D150 4100
?
E VON D152 BIS D1C5  $/+ $
D152 5352 $SRC
D154 4300
D156 4A4D $JMP
D158 5000
D15A 4A4C $JLT
D15C 5400
D15E 4A4C $JLE
D160 4500
D162 4A45 $JEQ
D164 5100
D166 4A48 $JHE
D168 4500
D16A 4A47 $JGT
D16C 5400
D16E 4A4E $JNE
D170 4500
D172 4A4E $JNC
D174 4300
D176 4A4F $JOC
D178 4300
D17A 4A4E $JNO
D17C 4F00
D17E 4A4C $JL
D180 0000
D182 4A48 $JH
D184 0000
D186 4A4F $JOP
D188 5000
D18A 5342 $SBO
D18C 4F00
D18E 5342 $SBZ
D190 5A00
D192 5442 $TB
D194 0000
D196 434F $COC
D198 4300
D19A 435A $CZC
D19C 4300
D19E 584F $XOR
D1A0 5200
D1A2 584F $XOP
D1A4 5000
D1A6 4C44 $LDCR
D1A8 4352
D1AA 0000
D1AC 5354 $STCR
D1AE 4352
D1B0 0000
D1B2 4D50 $MPY
D1B4 5900
D1B6 0000
D1B8 4449 $DIV
D1BA 5600
D1BC 2900 $)
D1BE 535A $SZC
D1C0 4300
D1C2 5300 $S
D1C4 2B00 $+
?
E VON D1C6 BIS D1F7  $/+ $
D1C6 4300 $C
D1C8 2C00 $,
D1CA 4100 $A
D1CC 2000 $
D1CE 4D4F $MOV
D1D0 5600
D1D2 534F $SOC
D1D4 4300
D1D6 403E $@>
D1D8 0000
D1DA 2852 $(R
D1DC 0000
D1DE 2052 $ R
D1E0 0000
D1E2 2A52 $*R
D1E4 0000
D1E6 2024 $ $>
D1E8 3E00
D1EA 2C3E $,>
D1EC 0000
D1EE 2024 $ $/+
D1F0 2F2B
D1F2 2000
D1F4 4E4F $NOP
D1F6 5000
?E VON D1F8 BIS D236  $/+
D1F8 C060 MOV @>FF02,R1
D1FA FF02
D1FC 8081 C R1,R2
D1FE 1B01 JH $>D202
D200 0380 RTWP
D202 C081 MOV R1,R2
D204 C0D1 MOV *R1,R3
D206 0283 CI R3,>4000
D208 4000
D20A 1A0C JL $>D224
D20C C103 MOV R3,R4
D20E 09D3 SRL R3,13
D210 0643 DECT R3
D212 0A23 SLA R3,2
D214 2FA3 XOP @>D1BE(R3),R14
D216 D1BE
D218 0A44 SLA R4,4
D21A 1702 JNC $>D220
D21C 2F20 XOP @>D0F2,R12
D21E D0F2
D220 0460 B @>D308
D222 D308
D224 0283 CI R3,>1000
D226 1000
D228 1366 JEQ $>D2F6
D22A 0A33 SLA R3,3
D22C 1813 JOC $>D254
D22E 0A13 SLA R3,1
D230 172A JNC $>D286
D232 C103 MOV R3,R4
D234 09C3 SRL R3,12
D236 0A23 SLA R3,2
?
E VON D238 BIS D2A6  $/+
D238 2FA3 XOP @>D156(R3),R14
D23A D156
D23C 0284 CI R4,>D000
D23E D000
D240 1402 JHE $>D246
D242 0460 B @>D394
D244 D394
D246 06C4 SWPB R4
D248 2FA0 XOP @>D100,R14
D24A D100
D24C 2E04 XOP R4,R8
D24E 09C4 SRL R4,12
D250 2E04 XOP R4,R8
D252 0380 RTWP
D254 0A13 SLA R3,1
D256 1812 JOC $>D27C
D258 C103 MOV R3,R4
D25A 09E3 SRL R3,14
D25C 0A23 SLA R3,2
D25E 2FA3 XOP @>D196(R3),R14
D260 D196
D262 0284 CI R4,>C000
D264 C000
D266 1402 JHE $>D26C
D268 0460 B @>D3A4
D26A D3A4
D26C 0A24 SLA R4,2
D26E 09C4 SRL R4,12
D270 0284 CI R4,>0009
D272 0009
D274 16F9 JNE $>D268
D276 05C2 INCT R2
D278 05C2 INCT R2
D27A 10F6 JMP $>D268
D27C 09E3 SRL R3,14
D27E 38C0 MPY R0,R3
D280 2FA4 XOP @>D1A6(R4),R14
D282 D1A6
D284 10F1 JMP $>D268
D286 0A13 SLA R3,1
D288 1709 JNC $>D29C
D28A 0283 CI R3,>8000
D28C 8000
D28E 1438 JHE $>D300
D290 09D3 SRL R3,13
D292 0A23 SLA R3,2
D294 2FA3 XOP @>D146(R3),R14
D296 D146
D298 0460 B @>D3B0
D29A D3B0
D29C 0A13 SLA R3,1
D29E 1708 JNC $>D2B0
D2A0 09C3 SRL R3,12
D2A2 38C0 MPY R0,R3
D2A4 2FA4 XOP @>D0F2(R4),R14
D2A6 D0F2
?
E VON D2A8 BIS D31A  $/+
D2A8 0208 LI R8,>0001
D2AA 0001
D2AC 0460 B @>D30A
D2AE D30A
D2B0 0A13 SLA R3,1
D2B2 1726 JNC $>D300
D2B4 0933 SRL R3,3
D2B6 0283 CI R3,>1100
D2B8 1100
D2BA 1A0C JL $>D2D4
D2BC C103 MOV R3,R4
D2BE 0A74 SLA R4,7
D2C0 0284 CI R4,>0000
D2C2 0000
D2C4 161D JNE $>D300
D2C6 0A43 SLA R3,4
D2C8 09D3 SRL R3,13
D2CA 0643 DECT R3
D2CC 38C0 MPY R0,R3
D2CE 2FA4 XOP @>D0CE(R4),R14
D2D0 D0CE
D2D2 0380 RTWP
D2D4 0A33 SLA R3,3
D2D6 09C3 SRL R3,12
D2D8 38C0 MPY R0,R3
D2DA 2FA4 XOP @>D098(R4),R14
D2DC D098
D2DE 0284 CI R4,>0019
D2E0 0019
D2E2 1B02 JH $>D2E8
D2E4 0460 B @>D3D0
D2E6 D3D0
D2E8 0284 CI R4,>0025
D2EA 0025
D2EC 1B02 JH $>D2F2
D2EE 0460 B @>D3E6
D2F0 D3E6
D2F2 0460 B @>D3F6
D2F4 D3F6
D2F6 2FA0 XOP @>D1F4,R14
D2F8 D1F4
D2FA 0380 RTWP
D2FC C060 MOV @>FF02,R1
D2FE FF02
D300 2FA0 XOP @>D0FB,R14
D302 D0FB
D304 2E91 XOP *R1,R10
D306 0380 RTWP
D308 04C8 CLR R8
D30A C151 MOV *R1,R5
D30C C185 MOV R5,R6
D30E 0246 ANDI R6,>000F
D310 000F
```

```
    D312 C1C5 MOV R5,R7
    D314 0A67 SLA R7,6
    D316 09C7 SRL R7,12
    D318 C245 MOV R5,R9
    D31A 0AA9 SLA R9,10
?
  E VON D31C BIS D38C  $/+
    D31C 09E9 SRL R9,14
    D31E C285 MOV R5,R10
    D320 0A4A SLA R10,4
    D322 09EA SRL R10,14
    D324 2F20 XOP @>D1CC,R12
    D326 D1CC
    D328 C249 MOV R9,R9
    D32A 131B JEQ $>D362
    D32C 0289 CI R9,>0002
    D32E 0002
    D330 130B JEQ $>D348
    D332 2FA0 XOP @>D1E2,R14
    D334 D1E2
    D336 C306 MOV R6,R12
    D338 06A0 BL @>D382
    D33A D382
    D33C 0289 CI R9,>0003
    D33E 0003
    D340 1A02 JL $>D346
    D342 2F20 XOP @>D1C4,R12
    D344 D1C4
    D346 1012 JMP $>D36C
    D348 05C2 INCT R2
    D34A 2FA0 XOP @>D1D6,R14
    D34C D1D6
    D34E 2E92 XOP *R2,R10
    D350 C306 MOV R6,R12
    D352 130C JEQ $>D36C
    D354 2FA0 XOP @>D1DA,R14
    D356 D1DA
    D358 06A0 BL @>D382
    D35A D382
    D35C 2FA0 XOP @>D1BC,R14
    D35E D1BC
    D360 1005 JMP $>D36C
    D362 2F20 XOP @>D1DF,R12
    D364 D1DF
    D366 C306 MOV R6,R12
    D368 06A0 BL @>D382
    D36A D382
    D36C 0588 INC R8
    D36E 0288 CI R8,>0002
    D370 0002
    D372 1305 JEQ $>D37E
    D374 C24A MOV R10,R9
    D376 C187 MOV R7,R6
    D378 2F20 XOP @>D1C8,R12
    D37A D1C8
    D37C 10D5 JMP $>D328
    D37E 0380 RTWP
    D380 C307 MOV R7,R12
    D382 0205 LI R5,>000A
    D384 000A
    D386 814C C R12,R5
    D388 1A03 JL $>D390
    D38A 2F20 XOP @>D0A2,R12
    D38C D0A2
?
  E VON D38E BIS D3FE  $/+
    D38E 6305 S R5,R12
    D390 2E0C XOP R12,R8
    D392 045B B *R11
    D394 C171 MOV *R1+,R5
    D396 0A85 SLA R5,8
    D398 0875 SRA R5,7
    D39A A141 A R1,R5
    D39C 2FA0 XOP @>D1E6,R14
    D39E D1E6
    D3A0 2E85 XOP R5,R10
    D3A2 0380 RTWP
    D3A4 04C8 CLR R8
    D3A6 C151 MOV *R1,R5
    D3A8 0245 ANDI R5,>F3FF
    D3AA F3FF
    D3AC 0460 B @>D30C
    D3AE D30C
    D3B0 C311 MOV *R1,R12
    D3B2 024C ANDI R12,>000F
    D3B4 000F
    D3B6 C1D1 MOV *R1,R7
    D3B8 0A87 SLA R7,8
    D3BA 09C7 SRL R7,12
    D3BC 2FA0 XOP @>D1DE,R14
    D3BE D1DE
    D3C0 06A0 BL @>D382
    D3C2 D382
    D3C4 2F20 XOP @>D1C8,R12
    D3C6 D1C8
    D3C8 C307 MOV R7,R12
    D3CA 06A0 BL @>D382
    D3CC D382
    D3CE 0380 RTWP
    D3D0 C332 MOV *R2+,R12
    D3D2 024C ANDI R12,>000F
    D3D4 000F
    D3D6 2FA0 XOP @>D1DE,R14
    D3D8 D1DE
    D3DA 06A0 BL @>D382
    D3DC D382
    D3DE 2FA0 XOP @>D1EA,R14
    D3E0 D1EA
    D3E2 2E92 XOP *R2,R10
    D3E4 0380 RTWP
    D3E6 C311 MOV *R1,R12
    D3E8 024C ANDI R12,>000F
    D3EA 000F
    D3EC 2FA0 XOP @>D1DE,R14
    D3EE D1DE
    D3F0 06A0 BL @>D382
    D3F2 D382
    D3F4 0380 RTWP
    D3F6 2FA0 XOP @>D100,R14
    D3F8 D100
    D3FA 05C2 INCT R2
    D3FC 2E92 XOP *R2,R10
    D3FE 0380 RTWP
?
```

fehls. Nach Eingabe dieser Adresse meldet er sich wieder mit:

E VON C000 BIS C060 $/+

Mit der Eingabe eines bestimmten Abschlußzeichens kann man dem Disassembler nun noch mitteilen, wie der spezifizierte Programmteil zu interpretieren ist:

a) ASCII-Text: Abschlußzeichen $
b) Hex-Zahlen: Abschlußzeichen +
c) Maschinenbefehle: beliebiges anderes Abschlußzeichen

Der Ausdruck hat folgendes Format:

Adresse – Maschinenbefehl – mnemonischer Code

Nach erfolgtem Ausdruck meldet der Computer sich wieder mit „?", und man kann das Programm erneut mit E starten.

Stichworte zum Inhalt

TMS 9900, TMS 9980, 16-bit-CPU, TM 990/302.

Einsteigen bitte...

...sonst fährt der Zug ab! Für Mikrocomputer-Einsteiger sind in der FUNKSCHAU und in unseren Sonderheften zahlreiche Beiträge erschienen, damit Sie nicht den Anschluß verpassen. Unser Mini-Literaturverzeichnis sagt Ihnen, wo wann worüber, und beantwortet damit zahlreiche schriftliche und telefonische Anfragen an die Redaktion.

Inhalt	*Titel*	*Publikation*
Funktionsweise von Mikrocomputern	Mikrocomputer – eine Einführung in kleinen Schritten	Sonderheft der FUNKSCHAU, DM 9.50
Maschinensprache-Programmierung	Z-80-Applikationsbeispiele	Franzis-Buch ISBN 3-7723-6671-6, DM 32.-
	Anwendungsbeispiele für den Mikroprozessor 6502	Franzis-Buch, RPB 173, ISBN 3-7723-1731-6, DM 8.80
	Bits und Bytes (6502-Programmierung)	Beitrag in „Hobbycomputer 2", Sonderheft der FUNKSCHAU, DM 14.–
Programmieren in Basic	Basic vor dem Handbuch	Serie in der FUNKSCHAU, Hefte 19...22/1980
	Basic für Microcomputer: Geräte, Begriffe, Befehle, Programme	Franzis-Buch, ISBN 3-7723-6821-2, DM 26.–

Sollten Sie das eine oder andere Heft, Sonderheft oder Buch einmal nicht im Handel erhalten, so wenden Sie sich bitte direkt an den Franzis-Verlag, Postfach 37 01 20, 8000 München 37.

Hans-Georg Joepgen

„Apple-Doc" befreit von Knochenarbeit

Bei der Entwicklung umfangreicher Basic-Programme nimmt die Buchführung über Variablen-Namen und Unterprogramm-Adressen schnell ein Maß an, das weitaus mehr Zeit und Aufmerksamkeit erfordert, als das eigentliche Programmverfassen, die Festlegung von Algorithmen und Abläufen nämlich. Aus Amerika kam nun ein Paket von Dienstleistungen für die Mikrocomputer Apple-II und ITT 2020, das den Programmierer von derlei Knochenarbeit weitgehend befreit: „Apple-Doc", ein Bündel höchst nützlicher Dokumentations- und Editierroutinen.

„Apple-Doc", verfaßt von Roger Wagner und vertrieben durch die Firma Southwestern Data Systems im kalifornischen Santee, wird zusammen mit einer vierzehnseitigen Bedienungsanleitung von befriedigender Qualität als Diskette oder Kassette verkauft (der Verfasser zahlte im Frühsommer 80 knapp 25 Dollar). Die Bedienung ist benutzerfreundlich gehalten, da, wo die Entscheidungen mehr Aufmerksamkeit erfordern, sind ausführliche Soufflier-tafeln eingebaut, die zur rechten Zeit auf dem Bildschirm erscheinen. Was dabei herauskommt, wenn man „Apple-Doc" seine Wünsche mitteilt, ist frappierend: Die Dienstleistungsroutinen erlauben, wahlweise auf Schirm oder Druckerpapier, die Ausgabe komfortabler Referenzlisten über die im Benutzerprogramm angesprochenen Variablen, Datenfelder und Funktionen, teilen mit, wo diese Variablen im einzelnen benutzt werden und wie oft pro Basic-Zeile. Weitere Listen erläutern, welche Zeilen woher durch THEN, GOTO oder GOSUB aufgerufen werden. Auf Wunsch führt Apple-Doc sogenannte Deskriptoren-Listen zu jedem Funktionen-, Feld-, Variablen- oder Zeilennamen: Aufzeichnungen mit Bemerkungen des Programmierers. Diese Listen können auf Diskette abgelegt werden und sind unter dem Namen des Benutzerprogramms, gefolgt von einem Suffix, jederzeit wieder zugänglich: Bis hierhin ein hochwillkommenes Dokumentationswerkzeug.

Robert Wagner läßt es aber nicht beim passiven Dokumentieren bewenden – er schuf zu diesen Dokumentationsroutinen noch eine Gruppe von Dienstleistungen zum aktiven Editieren. So können Variablen-Namen im Benutzerprogramm punktuell oder global geändert werden, beliebige Texte aufgesucht und ausgetauscht. Bei Programmkürzungen macht Apple-Doc den Programmierer gesondert darauf aufmerksam, daß die Zeile, die gerade entfernt werden soll, noch irgendwo als Ziel eines bedingten oder unbedingten Sprunges vermerkt ist: In Summe – wer jemals größere Basic-Programme verfaßt, sie modifiziert oder auch nur gründlich analysiert hat, für den ist klar, wieviel an Mühe ihm sein Rechner mit Apple-Doc-Hilfe abnehmen kann.

Wagners Trick: Versetzte Pointer

Die Floating-Point-Basicdialekte Palsoft und Applesoft, mit denen Wagners Dienstprogramme zusammenarbeiten, benutzen Pointer, mit denen Anfang und Ende von Text und Tabellen markiert werden. Mindestens einige der zahlreichen Apple-Doc-Routinen sind gleichzeitig mit dem Benutzer-Programm im Speicher – und sind doch selbst zu einem Gutteil in Applesoft geschrieben: Wie macht Wagner das? Durch (in der weniger komfortablen Kassettenversion) einige Immediate-Befehle oder (unter DOS 3.2) dem Kommando eines durch EXEC abgerufenen Textfiles „verbiegt" Wagner diese Pointer, schützt dadurch das im Speicher befindliche Benutzerprogramm und setzt seine Routinen im Anschluß daran in den RAM-Bereich. Das kostet Speicherplatz, jedoch ist durch zweierlei Möglichkeiten Vorsorge getragen, daß auch größere Benutzerprogramme bearbeitet werden können, ohne, daß sich die Programme wechselseitig wegen Platzmangel auf die Füße treten: Einmal läßt sich das Benutzerprogramm portionsweise analysieren, der innere Zusammenhang geht dadurch nicht verloren, zum zweiten hat Wagner bereits Möglichkeiten zur „Entfeinerung" einiger seiner Routinen vorgesehen. Durch Kürzen kann man Komfort gegen geringere Speicherbeanspruchung tauschen. Die Benutzungsanleitung sagt, wie.

Vorführung am praktischen Beispiel

Die Leistungsfähigkeit von Apple-Doc sei anhand eines im Frühjahr 80 mit komplettem Listing veröffentlichten Anwenderprogramms dargestellt – „Wobbelplotter", der Simulation eines Wobbelmeßplatzes durch ein PAL-SOFT-Programm, das den Frequenzgang von Netzwerken in farbigem Schaubild auf den Schirm plottet [1]. *Bild 1* zeigt für einen Ausschnitt aus „Wobbelplotter" die Variablentabelle. Dem Namen der Variablen, Funktion oder Matrix folgt der Descriptor und

```
*LG(*) - DEKADISCHER LOGARITHMUS
16

A - SAMMELVARIABLE "AKKU"
601 602 604 630 630 635 640 885 885 890 890 890 891 892 893 894
894 2160 2160 2160

A$ - SAMMEVARIABLE "STRINGAKKU"
14 14 120 120 120 120 666 667 667 668 669 670 672 1910 1912 1914
1916 1918 1920 1920 1920 1952 1952 1952 1954 1955 1960 1963 1963
1963 1964 1964 1964 1965 1975 1975 1976 1977 1978 1979 1980 1981
1982 2160 2160 2168 2168 2168

E10 - *** FALSCHE INTERPRETATION DER KONSTANTEN 1E10. APPLEDOC-FEHLER!
880
```

Bild 1. Ausschnitt einer Variablenliste, wie sie vom Apple-Doc-Programm „VRF" ausgedruckt wird. Man beachte die fälschlich ausgegebene und in Wahrheit nicht existente Variable E10 – aber das war auch der einzige Fehler, der Apple-Doc unterlief

sodann die Liste aller jener Zeilen, in denen das analysierte Programmsegment die Variable anspricht oder die Funktion benutzt. Hier wird übrigens auch ein Mißverständnis zwischen dem Apple-Doc-Programm „VRF" (Variablen-Referenzliste) und „Wobbelplotter" deutlich: Bei der angeblichen Variablen „E10", die „VRF" meldet, handelt es sich in Wahrheit um den verkürzten Rest der Konstanten 1E10 (1 x 10 hoch zehn) aus Zeile 880 von Wobbelplotter, wie *Bild 2* zeigt. *Bild 3* führt vor, wie eine vom Apple-Doc-Programm „LRF" (Zeilen-Refrenzliste) gefertigte Aufstellung aussieht. Der Zeilennummer folgt, sofern vom Operator in die Descriptorenliste einmal eingegeben, eine Erläuterung und danach die Aufzählung jener Zeilennummern, von wo aus ein Sprung oder ein Subroutinen-Aufruf zur anfangs gemeldeten Zeile erfolgt.

Ein Änderungsvorschlag

In der erprobten Originalversion von Apple-Doc erlauben VRF und LRF nicht die simultane Zusammenarbeit zwischen Datensenken unterschiedlicher Anforderungen, etwa Druckern verschiedener Zeilenbreite. Die Bedienungsanleitung gibt zwar präzise Hinweise, wie VRF und LRF durch Änderungen der Zeile 460 dem jeweils benutzten Druckertyp und seiner Zeilenbreite anzupassen sind, aber dies paßt nun ganz und gar nicht zum sonstigen Apple-Doc-Komfort. Vorgeschlagen wird die aus den *Bildern 4* und *5* ersichtliche Änderung, durch die VRF und LRF – die Änderung betrifft beide Programme gleichermaßen, da sie an dieser Stelle einschließlich der Zeilennummern deckungsgleich sind – in die Lage versetzt werden, ihre Druckausgaben unmittelbaren Benutzerwünschen anzupassen.

Schlußurteil

Das Programmpaket Apple-Doc, erprobt auf einem Computer ITT-2020 Palsoft und einem Apple-II-Plus mit jeweils 48 kByte RAM, beide mit DOS 3.2, hat – bis auf die erwähnte Fehlinterpretation der in Exponentialschreibweise formulierten Konstanten aus „Wobbelplotter" – in keinem Fall zu falschen Ergebnissen geführt. Nach wenigen Stunden war die Bedienung der Apple-Doc-Routinen dem Verfasser so „in die Finger" gegangen, daß sie nahezu automatisch erfolgte und die Bedienungsanleitung weggelegt werden konnte. Bereits nach kurzer Zeit wurde (fast) unvorstellbar, daß man zuvor beim Programmschreiben noch mit „Handbuchhaltung" gearbeitet hatte. Summa summarum: Apple-Doc zählt zu den besten Software-Käufen, die der Autor je tat.

Literatur

[1] *Joepgen, Hans-Georg:* Basic-Programm simuliert Wobbel-Meßplatz. ELEKTRONIK 1980, Heft 2, Seiten 49 ff.

[2] *Joepgen, Hans-Georg:* Wanzenjagd. Über die Fehlersuche in Basic-Programmen. „Hobbycomputer 2", Sonderheft der FUNKSCHAU. Franzis-Verlag, München.

Stichworte zum Inhalt

Apple-II, ITT-2020, Variablenliste, Zeilenreferenz-Tabelle.


```
870  PRINT "     F;(HZ)  D;(DB)      F;(HZ)  D;(DB)": GOSUB 885: RETURN
880 F =  INT (F + .5):F$ =  STR$ (F): IF F =  > 1E10 OR F < 1 THEN  PRINT
    : PRINT "FREQUENZ AUSSER BEREICH. BEENDET.": END
```

Bild 2. Teil aus dem im Januar 1980 veröffentlichten Programm „Wobbelplotter". In Zeile 880 taucht die Konstante 1E10 auf, die von VRF als Variablenname gedeutet wurde

```
  120 - EINGABE BAUTEILE: ART UND ANORDNUNG IM NETZWERK
120

  260 - EINGABE DER STARTFREQUENZ - RUECKSPRUNG BEI FEHLER
1916

  300 - HAUPTSCHLEIFE
669 1912 1914 2440
```

Bild 3. Ausschnitt einer Zeilen-Referenzliste, gefertigt vom Dienstprogramm LRF. Analysiert wurde wiederum der „Wobbelplotter"

```
]LIST450-460
450  HOME : INPUT "DO YOU WANT THE PRINTER ON? (Y/N)";I$:P1 =  PEEK (1912)
     :P2 =  PEEK (1913):P3 =  PEEK (1914):P4 =  PEEK (1915):P5 =  PEEK (19
     16):P6 =  PEEK (1917)
452  IF Q$ = "" THEN  PRINT : INPUT "SPECIAL FUNCTION? ('2*' OR 'RETURN'):
      ";Q$: PRINT
455  IF  LEFT$ (I$,1) <  > "Y" THEN 470
460 W = 40: PRINT  CHR$ (4);"PR#5": PRINT  CHR$ (9);"K": PRINT  CHR$ (9);W
    ;"N": PRINT  CHR$ (12): REM  W=WIDTH OF PRINTER OUTPUT; CHR$(12)= TOP
    -OF-FORM
```

Bild 4. So sieht der fragliche Teil von Apple-Doc ursprünglich aus – die Anpassung des Programms an unterschiedliche Drucker ist ziemlich umständlich

```
]LIST450-460
450  HOME : INPUT "DO YOU WANT THE PRINTER ON? (Y/N)";I$:P1 =  PEEK (1912)
     :P2 =  PEEK (1913):P3 =  PEEK (1914):P4 =  PEEK (1915):P5 =  PEEK (19
     16):P6 =  PEEK (1917)
452  IF Q$ = "" THEN  PRINT : INPUT "SPECIAL FUNCTION? ('2*' OR 'RETURN'):
      ";Q$: PRINT
455  IF  LEFT$ (I$,1) <  > "Y" THEN 470
456  PRINT : INPUT "# OF CHARS PER LINE (10..180) ?";W: REM   *** LINES 4
     56 AND 457 INSERTED , IN  LINE 460 "W=. .:" WAS TILT (JOE,  JULY 11TH,
     1980) ***
457  PRINT : IF (W < 10) OR (W > 180) THEN  PRINT  CHR$ (7)" +++ RANGE ERR
     OR": GOTO 456
460  PRINT  CHR$ (4);"PR#5": PRINT  CHR$ (9);"K": PRINT  CHR$ (9);W;"N": PRINT
      CHR$ (12): REM   W=WIDTH OF PRINTER OUTPUT; CHR$(12)= TOP-OF-FORM
```

Bild 5. Erhöhter Komfort bei Apple-Doc's Druckerausgabe. Die Änderung betrifft die Routinen VRF und LRF gleichermaßen

Formel-INPUT beim AIM-65

Das hier wiedergegebene AIM-65/PC-100-Programm baut auf dem in [1] abgedruckten Plot-Programm von Michael Konz auf. Neu ist die Möglichkeit, die zu untersuchende Funktion bei laufendem Programm eingeben zu können; dabei wurde das in [2] für den PET-2001 angegebene Verfahren verwendet und an den AIM-65 angepaßt.

Voraussetzung für die INPUT-Übernahme einer Formel ist die genaue Kenntnis der Abkürzungen, die der Computer intern für bestimmte Befehle verwendet. Ausführlich sind diese Dinge in [3] dokumentiert.

Unser Programm (*Bild 1*) setzt voraus, daß die „DEF FN"-Zeile die erste überhaupt ist; nur dann stimmen die angegebenen POKE-Adressen des AIM-65. Es ist also nicht zulässig, etwa in die Zeilen 1...4 noch REMs zu schreiben. Wie in [1] ist es auch nötig, eventuell den AIM-Trimmer VR 3 so einzustellen, daß ein sauberer, unverzerrter Kurvenausdruck zustande kommt. In *Bild 2* ist nochmals das hierfür ebenfalls nötige Maschinenprogramm angegeben, *Bild 3* zeigt einen Beispielausdruck.

Bei der Eingabe des Gesamtprogramms ist wie folgt vorzugehen:

1. Basic-Interpreter mit Memory Size = 3072 initialisieren und den Basic-Teil (Bild 1) eingeben.
2. Mit Escape aus Basic „aussteigen" und den Maschinenprogrammteil (Bild 2) eintippen.
3. Mit Taste 6 wieder in den Basic-Interpreter gehen und das Programm mit RUN starten. Hierbei sollte der Drucker ausgeschaltet sein; das Programm sorgt automatisch für den Ausdruck an den richtigen Stellen.

Sinnvollerweise geht man beim Abspeichern des Programms auf Kassette so vor, wie das ebenfalls in [3] beschrieben ist: Man ermittelt den Inhalt der Zellen 0076 und 0075. Die darin stehende Adresse merkt man sich und bringt mit dem Monitor-Befehl D zunächst den Bereich 0200 bis zu dieser Adresse auf Band. Nach MORE? gibt man Y und den Adressenbereich 0000...00D6 ein. Wenn wieder MORE? erscheint, läßt man nach Y noch den

Bild 1. Basic-Programm zum Plotten einer Funktion, die mit INPUT eingegeben wird. Zeile 99 sorgt dafür, daß kein „Error" bei Funktionen auftritt, die für X=0 nicht definiert sind (wie die Funktion in Bild 3). Die Zahl der Doppelpunkte 50 in Zeile 5 ist genau einzuhalten

```
<L>IN=T F=PLOT2 T=1
<6>
LIST

 5 DEFFNY(X)=::::::::::::::::::
 ::::::::::::::::::::::::::::::::
10 IFJP=0THENDIMTK$(23):GOTO10060
11 DEFFNZ(X)=1/FNY(X)
15 PRINT!" INTERVALLGRENZEN:
16 INPUT"X-MIN=";A:PRINT!"X-MIN=";A
17 INPUT"X-MAX";B:PRINT!"X-MAX=";B
20 IFA>BTHENPRINT"IRRTUM":GOTO16
30 E=(B-A)/100
35 N=-A/E:P=N
40 IF N<0 OR N>255 THEN P=255
50 POKE 3328,P
60 INPUT"Y-MIN";C:PRINT!"Y-MIN=";C
65 INPUT"Y-MAX";D:PRINT!"Y-MAX=";D
70 IFC>DTHENPRINT"IRRTUM":GOTO60
80 F=(D-C)/100:M=-C/F:P=M
84 IF M<0 OR M>255 THEN P=255
86 POKE 3324,P
88 PRINT!" ":PRINT"EINEN MOMENT BITTE
90 FOR X=A TO B STEP E
99 IFX=0THEN110
100 Y=FNY(X)
110 IF Y<C OR Y>D THEN Y=(255-M)*F
120 POKE INT(X/E+N+.5)+3840,INT(Y/F+M+.5)
130 NEXT
140 POKE4,0:POKE5,12:X=USR(0):PRINT!" "
170 PRINT"ANDERES INTERVALL?"
175 GETN$:IFN$=""GOTO175
180 IFN$="J"GOTO15
190 PRINT"NEUE FUNKTION?"
200 GETN$:IFN$=""GOTO200
210 IFN$="J"THENGOSUB10400:RUN
220 GOSUB10400:END
10060 IFPEEK(541)<>58THENJP=1:GOTO5
10080 INPUT"Y=";FC$:KL=541
10085 PRINT!" ":PRINT!"GEGEBENE FUNKTION:"
:PRINT!"Y="FC$:PRINT!" "
10086 RESTORE
10087 FORI=0TO22:READTK$(I):NEXT
10090 FORI=1TOLEN(FC$)
10100 FC%=ASC(MID$(FC$,I,1))
10105 FORJ=0TO22
10110 IFMID$(FC$,I,LEN(TK$(J)))=TK$(J)
THENFC%=J+164
10115 NEXTJ
10290 IFFC%>173ANDFC%<188THENI=I+2
10300 POKEKL,FC%:KL=KL+1:NEXTI:RUN
10400 FORI=541TO590:POKEI,58:NEXT:RETURN
11000 DATA+,-,*,/,^,#,#,#,#,#,SGN,INT,
ABS,USR,#,#,SQR,RND,LOG
11010 DATAEXP,COS,SIN,TAN,ATN
```

```
0C00 A9 FF 8D 74 | 0C34 01 A8 AD 00 | 0C68 C8 C0 0A F0 | 0C9C 20 AF 0C A2 | 0CD0 BD 00 0F CD
0C04 A4 A9 63 8D | 0C38 A8 0D 79 A4 | 0C6C 04 C0 14 D0 | 0CA0 0A 20 A0 FF | 0CD4 77 A4 D0 20
0C08 76 A4 A9 63 | 0C3C 8D 00 A8 A9 | 0C70 0A CE 77 A4 | 0CA4 CA 10 FA A9 | 0CD8 AD 78 A4 0D
0C0C 8D 77 A4 20 | 0C40 A4 8D 08 A8 | 0C74 AD 77 A4 C9 | 0CA8 E1 8D 0C A8 | 0CDC 7A A4 8D 78
0C10 BD 0C A9 C1 | 0C44 A9 08 8D 09 | 0C78 FF F0 20 20 | 0CAC 4C CB F0 A9 | 0CE0 A4 AD 79 A4
0C14 8D 0C A8 20 | 0C48 A8 AE 76 A4 | 0C7C BD 0C 20 1B | 0CB0 00 8D 01 A8 | 0CE4 0D 7B A4 8D
0C18 A0 FF D0 08 | 0C4C 2C 74 A4 10 | 0C80 EC 20 AF 0C | 0CB4 AD 00 A8 29 | 0CE8 79 A4 0E 7A
0C1C 20 A0 FF D0 | 0C50 0B CA E0 59 | 0C84 C0 14 F0 10 | 0CB8 FC 8D 00 A8 | 0CEC A4 2E 7B A4
0C20 03 4C 79 F0 | 0C54 D0 0F EE 74 | 0C88 A9 80 8D 08 | 0CBC 60 A9 00 8D | 0CF0 8A 38 E9 0A
0C24 A0 00 8C 01 | 0C58 A4 E8 D0 09 | 0C8C A8 A9 09 8D | 0CC0 7B A4 8D 78 | 0CF4 AA B0 D9 60
0C28 A8 A9 02 2C | 0C5C E8 E0 64 D0 | 0C90 09 A8 20 1B | 0CC4 A4 8D 79 A4 | 0CF8 AD 77 A4 C9
0C2C 0D A8 F0 FB | 0C60 04 CE 74 A4 | 0C94 EC 4C 30 0C | 0CC8 A9 01 8D 7A | 0CFC 00 F0 D9 E0
0C30 AD 78 A4 8D | 0C64 CA 8E 76 A4 | 0C98 4C 24 0C EA | 0CCC A4 AE 76 A4 | 0D00 32 F0 D5 D0
                                                                  0D04 E5
```

Bild 2. Das zum Plotten erforderliche Maschinenprogramm aus [1] ist hier der Vollständigkeit halber nochmals abgedruckt

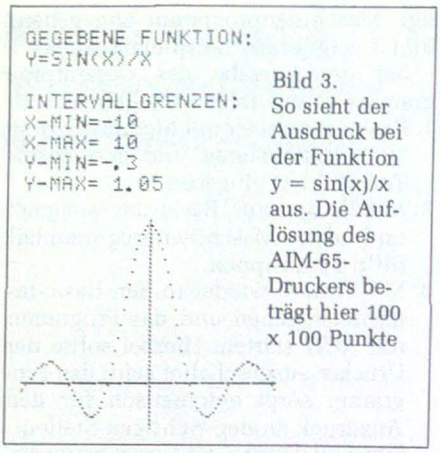


Bild 3. So sieht der Ausdruck bei der Funktion y = sin(x)/x aus. Die Auflösung des AIM-65-Druckers beträgt hier 100 x 100 Punkte

Bereich 0C00...0D04 folgen. Nach dem dritten MORE? drückt man N. Das Wiedereinlesen erfolgt mit der Monitorroutine „L", und mit Taste 6 gelangt man dann in Basic.

Jörg Struve, Herwig Feichtinger

Literatur

[1] AIM-65 plottet Funktionen. FUNKSCHAU 1980, Heft 12, S. 94.

[2] Formeleingabe mit INPUT. FUNKSCHAU 1980, Heft 2, Seite 77.

[3] Basic für Mikrocomputer. Franzis-Verlag, München, 1980. ISBN 3-7723-6821-2.

ROM und RAM im AIM-Assembler

Der für die Mikrocomputer AIM-65 und PC-100 erhältliche symbolische 4-KByte-Assembler enthält einige interessante Unterprogramme, die sich in eigenen Programmen oft gut verwenden lassen. Ferner ist es manchmal nützlich, zu wissen, welche RAM-Adressen vom ihm verwendet werden. Nachfolgend bringen wir daher eine grobe Übersicht von RAM- und ROM-Adressen.

Zero-Page-Adressen

00...03	Unbenutzt
04...0A	benutzt für Zeiger usw.
0B, 0C	Symbolanzahl in der Symboltabelle
0D, 0E	Opcode-Adresse in der ROM-Tabelle
0F...19	Verschiedene Variable
1A, 1B	Error-Anzahl
1C, 1D	Operanden-Variable
1E	Error-Zahl (Pass 2)
1F	Flag für Mehrfachzeilen
20, 21	Variablen
22	Flag für Assembler-Direktiven
23	Pass-Nummer
24...30	String-Start, -Ende, -Länge, Zahlenwert, Kolumne, Symbol, Operandenlänge
31	Error-Codezahl
32, 33	Programmzähler
34	Kolumnenzähler
35...39	Verschiedene Variablen
3A, 3B	Startadresse der Symboltabelle
3C, 3D	Symboltabellen-Adresse
3E, 3F	Endadresse der Symboltabelle
40, 43	Prüfvariablen
44, 45	Programmzähler für Listing
46...88	Verschiedene Variablen
89...A6	Ausgabe-Pufferspeicher
A7...AB	Speicher für Input-Filename
AC...FF	Unbenutzt

(Der Stackbereich wird von 0170 bis 0183 als Code-Buffer benutzt.)

ROM-Adressen

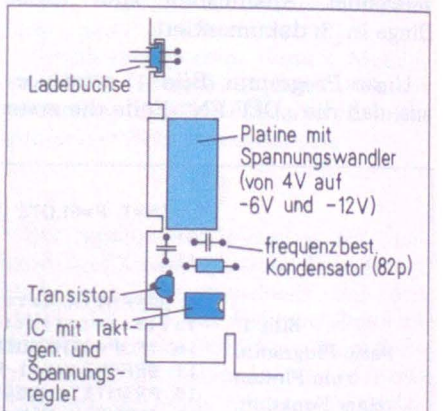
D000	Startadresse des Assemblers
D052	Option-Flags auf Normalwert setzen
D0BE	„Pass 1" drucken
D0C3	Pointer setzen, Variablen löschen
D172	Verarbeiten der Labels in Pass 1
D1DB	Behandlung von „="
D259	Decodieren von Assembler-Direktiven
D396	Behandlung von .OPT
D454	Operanden-Verarbeitung (inkl. Adressierungsart)
D690	Behandlung von .FIL

Unterprogramme

D6CA	Überlesen von Leerräumen
D6E8	String-Ende suchen
D720	Komma oder geschlossene Klammer suchen
D76B	String in Anführungszeichen ausgeben
D75C	Zeichentest: C=1, wenn Buchstabe
D768	Zeichentest: C=1, wenn Ziffer
D774	Erstellen eines Symbols
D797	Berechnen des Wertes eines Ausdrucks
D7EF	Länge eines Symbols feststellen
D8C3	C=1 setzen, wenn ein Leerraum, ein Komma, eine geschlossene Klammer oder ein Semikolon gefunden wird
D8DB	Übersetzen des komprimierten Mnemonic-Codes
D8ED	Umwandeln einer oktalen, dezimalen oder hexadezimalen ASCII-Zahl in eine Binärzahl
D95E	Absuchen der Symboltabelle nach einem Namen
D9A2	Neues Symbol in die Symboltabelle aufnehmen
D9CC	Symboltabellen-Überlauf
D9D4	Opcode-Tabelle absuchen
DA1B	Relative Sprungadressen korrigieren
DA5E	Eine Zeile ausdrucken
DBC7	Error-Zeilen ausdrucken
DD4B	Tabelle mit Sprungadressen
DD75	ASCII-Tabelle mit Mnemonics von Direktiven und Befehlen
DE5C	Konstantentabellen
DF4E	ASCII-Tabellen mit Texten
DFA8	Auf .END oder .FIL prüfen
DFFA	Leerraum ausgeben

Schnellerer TI-58

Der Taktoszillator des Rechners TI-58 ist in einem 8-Pin-IC auf der Platine enthalten, das auch einen Spannungsregler enthält. Frequenzbestimmend ist ein Keramik-Kondensator, der im Originalzustand einen Wert von 82 pF besitzt.



Durch Verringern des frequenzbestimmenden Kondensators läßt sich bei manchen programmierbaren Rechnern, wie hier beim TI-58, eine Erhöhung der Geschwindigkeit erreichen

Die Taktfrequenz ist so gewählt, daß auch bei größeren Exemplarstreuungen noch keine Schwierigkeiten auftreten. Naheliegend ist es nun, diesen Kondensator so weit zu verringern, daß das Diagnose-Programm im Standard-Modul gerade noch keinen Fehler meldet. Im Mustergerät stellte sich heraus, daß bei einem Wert von 68 pF der Rechner noch absolut zuverlässig funktioniert.

Es ist also Sache des Benutzers, seinen Rechner auf die optimale Taktfrequenz zu trimmen. Jan Brinckmann

Anm. d. Red.: Dieser Trick funktioniert auch bei manchen anderen Rechnern, jedoch nicht beim TI-59, weil dieser mit einem Keramik-Resonator arbeitet. Eine Veränderung der TI-59-Taktfrequenz würde ferner zum Nichtfunktionieren des Magnetkarten-Lesers führen. Beim TI-58 oder HP-25 läßt sich durch diese Methode jedoch eine Geschwindigkeitserhöhung von rund 20 % erzielen.

Hans-Georg Joepgen

„Präsenztest" schützt vor Desaster

Verfahren zum Schutz gegen Datenverluste

In jüngster Zeit bürgert sich mehr und mehr eine neue Programmform ein, eine Art „Misch-Stil": Der Job selbst ist zwar weitgehend in einer höheren Programmiersprache formuliert, enthält jedoch wiederholt maschinensprachliche Segmente, die zur Ablaufbeschleunigung oder besseren Speicherplatzausnutzung aufgerufen werden. Von den spezifischen Gefahren, die mit einer derartigen mehrsprachigen Programmformulierung verbunden sind, und von einem Verfahren zu deren Bewältigung handelt der folgende Beitrag.

Neuzeitliche Rechner-Betriebssysteme tragen durch eine Fülle eingebauter Syntaxprüfungen und anderer Sicherheitsvorkehrungen dafür Sorge, daß Programmierfehler nicht zu Datenverlusten oder Diskettenbeschädigungen führen. Diese Sicherungen werden jedoch dann umgangen, sobald von einer Hochsprache aus unmittelbar sprachfremde Subroutinen aufgerufen werden: wenn sich ein Basic-Programm beispielsweise per USR oder CALL einer Assemblersequenz bedient. Steht die gerufene Maschinenroutine zum Zeitpunkt des Aufrufs „beschädigt" zur Verfügung, geschieht möglicherweise Übles: Das Betriebssystem wird neurotisch, verfälscht unter Umständen Datensätze, gerät vielleicht sogar völlig außer Kontrolle. Dergleichen erlebte unlängst der Verfasser, als ein außer Tritt geratenes Assembler-Prögrämmchen einen Folgefehler ins DOS des Rechners verschleppte, der dann den Verlust einer Diskettenaufzeichnung zur Folge hatte.

Abhilfe: Präsenzüberprüfung

Zur Sicherung vor einer Wiederholung dieses Vorfalls wurde nach einem Verfahren zur Überprüfung des Vorhandenseins und der Unversehrheit von Maschinenroutinen gesucht. Forderung dabei: Ergab die Überprüfung, daß mit der Routine etwas nicht in Ordnung war, sollte erst einmal versucht werden, sie selbsttätig von Diskette nachzuladen. Verlief eine neuerliche Überprüfung danach erneut mit negativem Ergebnis, sollte das Programm bereits vor dem Subroutinenaufruf CALL abgebrochen werden und eine entsprechende Mitteilung auf dem Bildschirm erscheinen. Für die Überprüfung selbst wurde der Name „Präsenztest" festgelegt und bestimmt, daß sie in einer Basic-Subroutine erfolgen solle. Für die Kommunikation zwischen Präsenztest und aufrufendem Hauptprogramm sollte eine Boole'sche Variable mit Namen PRESENT dienen, die folgender Definition unterlag: Wenn PRESENT ungleich Null, dann war die Überprüfung erfolgreich.

Flag-Gewinn durch Prüfsummenbildung

Zur Durchführung von „Präsenztest" bot sich zunächst die in Disketten- und Kassetten-Betriebssystemen oft verwendete XOR-Methode an, bei der die Bytes im Prüfbereich bitweise in steigender Folge einer Exklusiv-Oder-Verknüpfung unterworfen werden und das zuletzt gewonnene Prüfbyte mit seinem Soll-Wert verglichen wird. Entsprechende Versuche auf einem PET-2001 verliefen erfolgreich, brachten auf einer ITT-2020 und verschiedenen Apple-II-Versionen jedoch Schwierigkeiten zutage, weil auf diesen Maschinen von Basic aus bitweise Verknüpfung von Bytes nicht unmittelbar zur Verfügung gestellt wird. Die Suche nach einem auf allen Maschinen mit Microsoft-Basic-Dialekten anwendbaren Verfahren ergab schließlich, daß der Prüfzeichengewinn durch fortgesetzte Summenbildung ausreichende Sicherheit bietet. Theoretisch versagt Präsenztest in der schließlich gewählten Form nur dann, wenn

1. jede Bit-Verfälschung durch die hierzu korrespondierende Bitmuster-Störung kompensiert wird (ein fälschlich auf High gesetztes Bit der Stelle n und ein fälschlich auf Low gesetztes Bit der gleichen Stelle in einem anderen Byte);

2. ein Falsch-Maschinenprogramm zufällig zur gleichen Prüfsumme führt wie das Nutzprogramm.

Wahrscheinlichkeits-Überlegungen zeigen, daß die zu erwartende Fehlerquote so gering ist, daß sie in der Praxis in den meisten Fällen sicherlich in Kauf genommen werden kann.

„Präsenzflag" am praktischen Beispiel

Zur Demonstration von Präsenzflag PRESENT und Präsenztest wurde ein Programm gewählt, das auf den Maschinen der Apple-Klasse läuft und das Betriebssystem des Mikrocomputers so modifiziert, daß die laufenden Benutzerprogramme jeweils zur vollen Sekunde unterbrochen werden und die in einem geschützten Feld auf dem Schirm erscheinende Datums- und Uhrzeitanzeige aktualisiert wird. Interrupt-Anforderung und Daten hierzu kommen von der Zusatzkarte „Apple Clock" (Mountain Hardware, Santa Cruz, Kalifornien), die hier im Steckplatz 4 der 2020 montiert wurde. Das PALSOFT-Programm „IRQ-UHR" (*Bild 1*) bedient sich einer Gruppe in Assembler geschriebene Routinen (Autor Sheri Talbott), die in Seite 3 des RAM-Bereichs abgelegt werden.

Das Programm „IRQ-UHR" stellt in den Zeilen 21...29 eine Texttafel („MENUE") zur Verfügung und bietet seine Dienstleistungen an. Stellt sich bei Erledigung des jeweils gewählten Auftrags heraus, daß hierbei auf eine Routine aus Sheri Talbott's Dienstleistungssammlung „CLOCK INTERRUPT.BIN" zurückgegriffen werden

muß, führt „IRQ-UHR" einen Präsenztest durch und versucht gegebenfalls, sich das defekte Binärfile erneut von der Diskette zu beschaffen.

Gewinn der Prüfsumme bei der Programmentwicklung

Der Programmierer muß die Prüfsumme der Maschinenroutinen kennen. Die Ermittlung erfolgt nach Laden des Maschinensprache-Blocks durch eine einzige Immediate-Kommando-Eingabe:

START = < Startadresse dezimal>: SCHLUSS = < Schlußadresse dezimal>: INHALT = Ø: FOR N = Ø TO SCHLUSS: INHALT = INHALT + PEEK (START + N): NEXT: PRINT INHALT

„Präsenzflag" und „Präsenztest" haben beim Verfasser voll zufriedenstellende Betriebssicherheit gezeigt. Es kam in keinem Fall mehr zu einem Verlust der Kontrolle über den Rechner. Wohl aber zeigte einige Male das Anlaufen einer Diskettenstation, daß „Präsenztest" tatsächlich auf ein gestörtes Maschinensprache-Programm gestoßen war und den Fehler selbsttätig behoben hatte. Zu dem in Microsoft-Basic-Dialekten sonst unüblichen Einsatz unabgekürzter Variablennamen sei auf das in (2) geschilderte Verfahren „Variablenwächter" verwiesen.

```
]LIST
10  REM ************************
11  REM *                      *
12  REM *       IRQ-UHR        *
13  REM *       -------        *
14  REM *                      *
15  REM *    10.7.80/JOE       *
16  REM *                      *
17  REM ************************
18  REM
19  REM MENUE
20  REM =====
21  HOME : PRINT "INTERRUPT-UHR      10.7.80/JOEPGEN":
    PRINT "KARTE: SLOT # 4": PRINT "---------------"
22  PRINT " ": PRINT " ": PRINT "+ DIENSTROUTINE NUR LADEN
    . . (1)": PRINT
23  PRINT "+ UHR UND DATUM EINBLENDEN. . (2)": PRINT
24  PRINT "+ NORMALBETRIEB. . . . . (3)": PRINT
25  PRINT "+ UHR STOP. . . . . (4)": PRINT
26  PRINT "+ UHR WIEDERANLAUF. . . . . (5)": PRINT
27  PRINT "+ ENDE. . . . . (6)": PRINT
28  FOR N = 0 TO 31: PRINT "-";: NEXT : PRINT "-"
29  REM
30  REM KOMMANDO
31  REM ========
32  PRINT : VTAB 20: PRINT "+++ KOMMANDO              (1..6)?";
33  GET A$: IF (A$ < "1") OR (A$ > "6") THEN  PRINT  CHR$ (7);:
    GOTO 33
34 AUFTRAG =  VAL (A$) - 1
35  REM
36  REM AUFTRAEGE 1 UND 6
37  REM ==================
38  IF AUFTRAG = 5 THEN  HOME : END
39  GOSUB 60: REM  PRAESENZTEST
40  IF  NOT PRESENT THEN  GOSUB 68: REM  LADEN
41  IF  NOT PRESENT THEN A$ = "LADEVERSUCH ERFOLGLOS": GOTO 53
42  REM
43  REM AUFTRAEGE 2..5
44  REM ===============
45  IF  NOT AUFTRAG THEN A$ = "OK. -              ": GOTO 52
46  RESTORE : FOR N = 1 TO AUFTRAG: READ ADRESSE: NEXT
47  CALL ADRESSE:AUFTRAG = 0: GOTO 45
48  DATA 822,849,869,871
49  REM
50  REM VOLLZUGSMELDUNG
51  REM ===============
52 A$ = "OK. -                "
53  PRINT  CHR$ (7): VTAB 21
54  PRINT "*** "A$;
55  FOR N = 1 TO 750: NEXT : PRINT  CHR$ (7);: IF PRESENT GOTO 21
56  GOTO 32
57  REM
58  REM PRAESENZTEST
59  REM ============
60 ADRESSE = 768:INHALT = 0
61  FOR N = 0 TO 104
62 INHALT = INHALT +  PEEK (ADRESSE + N)
63  NEXT :PRESENT = (INHALT = 12581)
64  RETURN
65  REM
66  REM LADEN
67  REM =====
68  ONERR  GOTO 71
69  PRINT : PRINT  CHR$ (4)"BLOAD CLOCK INTERRUPT.BIN"
70  GOSUB 60
71  RETURN
72  REM ************************
```

Datum und Zeit sekundengenau im Blickfeld: Dieses Basic-Programm verändert das Betriebssystem von Mikrocomputern wie Apple-II und ITT-2020 so, daß ein geschütztes Datenfeld auf dem Bildschirm ständig für die alle Sekunden per Interrupt erfolgende Datenausgabe der Apple-Clock-Karte reserviert bleibt – ein Beispiel für Datensicherung mit „Präsenztest"

Literatur

[1] *Joepgen, Hans-Georg:* „Bit-Flags ermöglichen elegante Basic-Wege" in „Programme für Kleincomputer und Taschenrechner", Sonderheft Nr. 31 der FUNKSCHAU. Franzis, München.

[2] *Joepgen, Hans-Georg:* „Variablen-Wächter sorgt für mehr Programm-Transparenz", ELEKTRONIK 1980, Heft 8, Seite 87, Franzis, München.

[3] Mountain Hardware, Inc. (Herausgeber): „Apple Clock Operating Manual", 2nd revised Edition July 1979, 300 Harvey West Blvd, Santa Cruz, CA 95060, USA

Stichworte zum Inhalt

Maschinensprache, 6502, Floppy-Disk, Apple-II, ITT-2020, Uhr

Inkompatibel?!

Maschinenprogramme und sogar manche Basic-Programme (speziell solche mit PEEKs und POKEs) sind systemspezifisch. Benutzt man einen anderen Computer als der Programmentwickler, wird ein Umschreiben erforderlich.

Für Computer mit der CPU 6502 ist das kein Kunststück mit unserem vor kurzem erschienenen Sonderheft „Mikrocomputer-Anwendungen": Es enthält ROM- und RAM-Tabellen für KIM-1, AIM-65, PET und CBM.

Jürgen Stollenwerk

Der Selbstbau-Computer

Erfahrungen mit dem Computer H-89 und dem Drucker H-14

Der Computer H-89 von Heathkit arbeitet mit zwei Z-80-Prozessoren und wird als Bausatz angeboten (siehe Heft 25/1979, S. 1473). Lesen Sie, was passiert, wenn der Bausatz nicht auf Anhieb funktioniert, und ob sich der Selbstbau lohnt.

Bausatz komplett

Der Computer H-89 kam in einem großen Pappkarton, die Einzelteile wohl verpackt, geordnet und mit einem Lageplan der Baugruppen versehen. Es fehlte nichts, bis zur letzten Unterlegscheibe war alles vorhanden!

Die Baumappen waren schnell zur Hand und dank der sehr ausführlichen Beschreibung (leider nur in Englisch) waren die Tüten mit den jeweiligen Bauteilen leicht zu finden und die Bauteile zu bestimmen. Die Lötarbeiten konnten beginnen. Es zeigte sich, daß die mitgelieferte Checkliste, die Punkt für Punkt abgehakt wurde, eine sehr hilfreiche Einrichtung war.

Neben dem Auflöten der elektronischen Bauteile auf die Netzteil- und Videoplatine, bei denen es keine besonderen Schwierigkeiten gab, ist der mechanische Zusammenbau zu erwähnen. Auch hier gab es keine besonderen Probleme, denn auch hier führte die Baumappe in Verbindung mit großen Detailzeichnungen Schritt für Schritt durch die notwendigen Arbeiten.

Das Herz des Computers H-89 sind die beiden großen Platinen, bestehend aus CPU und Terminallogik. Sie sind schon fertig bestückt und geprüft und brauchen nur noch eingehängt, verschraubt und mit Steckverbindungen versehen zu werden. Ein wohltuender Vorgang, wenn man die vielen ICs und Lötstellen auf engstem Raum betrachtet.

Eine der wichtigsten Arbeiten am Zusammenbau, ist die Verdrahtung des Gerätes. Hier muß sorgfältigste Arbeit geleistet werden, denn es sind eine Reihe von Steckverbindungen herzustellen, die eine Vielzahl von Einzelanschlüssen notwendig machen. Hierzu gleich ein Hinweis: Steckverbindungen sind immer problematisch! Deshalb ist hier besondere Sorgfalt geboten; man muß sich wirklich Zeit lassen.

Der Vollständigkeit halber sei noch erwähnt, daß neben den Steckverbindungen noch eine Reihe von einzelnen Drahtverbindungen herzustellen sind und ein Kabelbaum, schon fertig gebunden, eingelötet werden muß. Letztlich trägt er für ein einigermaßen aufgeräumtes Aussehen im Gerät bei. Von Drahtverhau, wie oft bei ausländischen Geräten zu sehen, kann man hier nicht sprechen, obwohl noch Verbesserungen möglich wären.

Reparatur im Werk

Nach einer Bauzeit von etwa einer Woche war das Gerät (ohne Floppy) fertig erstellt. Nach den letzten Checks, die keinen Fehler vermuten ließen, wurde das Gerät erstmalig eingeschaltet. Der eingebaute Lautsprecher gab einen kurzen Piepser von sich und das Gebläse ein unüberhörbares Rauschen als Zeichen dafür, daß es kräftig Luft umherwirbelte. Der Bildschirm aber blieb dunkel! Die Enttäuschung war groß. Eifriges Suchen begann – jedoch ohne Erfolg. Nach einigen Tagen erfolgloser Fehlersuche wurde der Computer verpackt und an Heathkit verschickt.

Nach etwa drei Wochen kam der H-89 zurück. Im Reparaturbericht stand: IC von Stromversorgung falsch eingebaut – neben einigen anderen kleineren Fehlern. Eindeutige Fehler durch unaufmerksamen Zusammenbau! Preis von Reparatur und Fracht hin und zurück etwa 150 DM. Voller Hoffnung und Sorge über den womöglich rauhen Transport des Gerätes erfolgte eine sofortige Funktionsprobe: Aus dem Lautsprecher ertönte ein zweifacher Piepser und nach kurzer Anheizzeit der Bildröhre, erschien oben links das ersehnte „H:" auf dem Schirm, womit der Computer anzeigte, daß er bereit sei und auf Anweisungen warte. Flugs wurde die fertig bestückte, verdrahtete und geprüfte Minifloppy eingebaut; eine Sache von wenigen Schrauben und Steckverbindungen.

Nach dem umfangreichen Manual (leider wieder nur in Englisch), das dem H-89 beiliegt, muß jede Diskette zuerst „initialisiert" das heißt, gekennzeichnet, werden. Aber das Initialisieren klappte nicht, obwohl es X-mal genau nach Anweisung versucht wurde. Nach mehreren Telefonaten mit Heathkit stand fest: Der H-89 war defekt, ehe daß er auch nur ein Programm abarbeiten konnte. Also wieder einpacken, versenden und warten.

Die Reparatur dauerte wieder rund drei Wochen. Der Reparaturbericht gab Auskunft über Kontaktfehler im Gerät und falschem IC in der fertig gelieferten Floppy (!). Die Reparaturkosten beliefen sich mit Fracht auf etwa 250 DM. Aber jetzt ging's los: Programme wurden erstellt, der Texteditor und die graphischen Zeichen ausprobiert. Alles funktionierte tadellos! Ein Schwachpunkt am Gerät ist aber zweifellos das Gebläse, das über der Stromversorgung angeordnet ist. Zu sehen ist es zwar nicht, aber dafür deutlich zu hören und für das stille Kämmerlein ein Nervtöter ersten Ranges. Als Nothilfe kann man zwar das Gebläse abstellen und zur Vermeidung eines Wärmestaus den Gerätedeckel abnehmen, das offene Gerät jedoch darf man dann nicht mehr aus den Augen lassen.

Auf Wunsch: Deutsche Tastatur

Inzwischen wurde von Fa. Heathkit ein Umbausatz für eine deutsche Tastatur zum Preis von etwa 300 DM angeboten. Dieser enthält einige ICs, leider nicht alle benötigten und bis heute nicht nachgelieferten Tastenköpfe zum Auswechseln, und eine einfache Anleitung zum Umbau. Auch dieser Umbausatz wurde angeschafft und in einer Viertelstunde ohne Schwierigkeiten eingebaut. Ganz unproblematisch ist das Arbeiten mit dem neuen Zeichengenerator jedoch nicht. Das Problem wurde nämlich dahingehend gelöst,

daß zugunsten der deutschen Umlaute in Klein- und Großbuchstaben und dem Buchstaben „ß" wenig gebräuchliche Zeichen einfach weggelassen wurden. Das wäre ja auch nicht so schlimm, wenn nicht das ursprünglich vorhandene Zeichen „@" durch „ß" ersetzt worden wäre. Unglücklicherweise ist aber das ursprüngliche Zeichen bei Programmen, geschrieben in Microsoft-Basic, belegt mit einer Funktion, die man etwa wie folgt interpretieren kann: Programmzeile zu lang – kann aber mit diesem Zeichen in der nächsten Zeile fortgesetzt werden. Dies führt dazu, daß bei Benutzung von „ß" bei „MBasic" der Cursor sofort an den Anfang der nächsten Zeile springt und dort verharrt. Der Buchstabe „ß" darf deshalb hierbei nicht benutzt werden.

Ein Telefonat mit Heathkit deckte auf, daß man dort über diese Zusammenhänge noch nicht Bescheid wußte, gab aber nach kurzer Bedenkzeit den Mangel zu und meinte, wohl achselzuckend: man müsse mit diesem Fehler eben leben!

In Verbindung mit dem Drucker H-14 sind die Probleme mit den deutschen Schriftzeichen noch schlimmer. Da der H-14 einen eigenen Zeichengenerator besitzt, der aber keine deutschen Schriftzeichen im ROM gespeichert hat, druckt er nach wie vor die ursprünglichen Zeichen aus, obwohl auf der Tastatur des Terminals und auf dem Bildschirm ein anderes Zeichen ausgewiesen ist. Auch mit diesem Mangel muß man wohl leben, denn ein neuer Zeichengenerator für deutsche Verhältnisse ist für den H-14 nach Auskunft von Heathkit nicht geplant.

Lohnt sich der Selbstbau?

Etwa vier Monate arbeitete der H-89 bis auf den Ventilator einwandfrei. Lediglich die Betriebssoftware erwies sich bei häufigem Diskettenwechsel als etwas umständlich, weil sie immer wieder neu geladen werden muß und das kostet Zeit. Ein Mangel, der bei einem geänderten Betriebssystem sicher behoben werden kann. In diesem Punkt ist der H-89 lobenswert flexibel.

Dann aber geschah es! Ohne erkennbaren Grund und mitten in einem fertigen Programm stieg der H-89 aus dem Programm aus, „vergaß" plötzlich das eingegebene Datum und die Floppy lief dauernd hin und her. Nach einigen Versuchen stieg er auch aus dem Betriebssystem aus und nahm letztlich überhaupt keine Befehle mehr an. Also wurde der H-89 wieder eingepackt und zu Heathkit eingeschickt. Die Reparatur dauerte runde fünf Wochen und kostete mit allem Drum und Dran etwa 430 DM. Auf dem Reparaturbericht stand; Speicher-IC defekt und Wärmefehler. Sogleich ausprobiert, zeigte sich, daß mehrere Disketten, die vorher einwandfrei im Gebrauch waren, nicht mehr angenommen wurden; sie waren durch das vorher defekte Gerät in Mitleidenschaft gezogen worden. Deshalb sei an dieser Stelle wärmstens empfohlen, von allen Programmen und Daten Kopien anzulegen, auf dem Laufenden zu halten und getrennt aufzubewahren. Bei Heathkit vorsichtig auf etwaige Garantie angesprochen, konstatierte man freimütig, daß das reparierte Gerät ein Selbstbaugerät und die Garantie bereits abgelaufen sei. Bedenkt man nun die Kosten, die im vorliegenden Fall entstanden sind, so ist die Ersparnis durch den Selbstbau schon fast aufgezehrt, nicht gerechnet die eigene Arbeitszeit und den Nutzungsausfall. Ob ein H-89 als Bausatz oder fertig gekauft wird, muß deshalb wohl überlegt sein, und wer sich da nicht sicher fühlt, sollte lieber das fertige Gerät ordern. Gewiß muß nicht jeder soviel Pech bei seinem Selbstbaugerät haben, aber rechnen muß man damit immer.

Der Drucker

Der Drucker H-14 ist ebenfalls ein recht interessantes Gerät. Er wurde auch als Bausatz angeschafft und zeigte ebenfalls in Verpackung, Baumappe (nur in Englisch) und Bauplan das schon erwähnte und hier nochmals gelobte, vorbildliche Muster der Vorbereitung und Anleitung zum Zusammenbau. So gab es auch hier keine besonderen Schwierigkeiten, und das Gerät arbeitete auf Anhieb fehlerfrei!

Besonders erwähnt werden sollte vielleicht das wirklich stabile Spritzgußunterteil, das alle elektrischen und mechanischen Bauteile aufnimmt. Das helle Oberteil aus Kunststoff läßt dagegen keine besondere mechanische Stabilität vermuten. Was die Druckqualität des Druckes angeht, so kann man damit wirklich zufrieden sein. Auch die gradlinigen Zeilen beeindrucken, was auf eine gute mechanische Führung des Schreibkopfes zurückzuführen ist. Lediglich die Zeilenlänge variiert manchmal bis zu einem halben Zeichen am Zeilenende. Nun fällt dies beim Ausdruck von geschriebenen Programmen nicht so sehr ins Gewicht, beim Ausdruck von Tabellen aber werden die Unterschiede deutlich: Die Zeichen stehen nicht genau untereinander.

Bei Heathkit wird an Verbesserungen am Computersystem fieberhaft gearbeitet. Leider ist der H-89 primär für den amerikanischen Markt gemacht. So muß man sich eben mit dem deutschen Bausatz für die Umlaute behelfen und auch die Beschreibung der Betriebssoftware, Benton-Harbor-Basic und Microsoft-Basic dazukaufen, allerdings recht teuer. So kosten die deutschen Handbücher immerhin zusammen ca. 140 DM. Ein stolzer Preis fürwahr! Eigentlich sollte ein Gerät, das auf dem deutschen Markt angeboten wird, von einer deutschen Beschreibung begleitet werden, aber wem diese Zusatzausgabe zuviel ist, der kann sich mit Englischkenntnissen immer noch mit der gelieferten englischen Ausgabe behelfen. Nur gut, daß der Hersteller des H-89 und des H-14 kein japanisches Unternehmen ist...

Alles in allem hinterläßt der Computer H-89 und der Drucker H-14 mit der komfortablen Software einen guten Eindruck. Das System bietet viel Variationsmöglichkeiten, und die beschriebenen Nachteile werden sicher durch die zunehmende Konkurrenz auf diesem Gebiet verbessert werden. Weil es den „perfekten Computer" ohnehin nicht geben kann, bieten der H-89 und der Drucker H-14 von Heathkit sicherlich eine gute Alternative.

Geknacktes Basic

Mehrmals hat die FUNKSCHAU bereits Programme veröffentlicht, mit denen sich der Wortschatz von Basic-Computern erweitern läßt. Wie funktioniert das überhaupt?

Um neue Befehlsworte decodieren zu können, muß man in den Basic-Interpreter eine zusätzliche Abfrage einbauen, die überprüft, ob das neu definierte Wort im Programm vorkommt und entweder wieder zum Interpreter zurück oder zu dem vom Anwender geschriebenen neuen Maschinenprogrammteil springt.

Bei allen bekannten 6502-Microsoft-Interpretern ist diese Abfrage dadurch zu bewerkstelligen, daß man die bei der Basic-Initialisierung in die Zero Page geladene Zeichenhol-Routine „knackt" und Verzweigungen zu eigenen Befehlen einbaut. Wir werden auf diesen Trick demnächst ausführlicher eingehen.

Super-Basic für AIM-65 und PC-100

Die Aachener Firma GWK hat für die Mikrocomputer AIM-65 von Rockwell und PC-100 von Siemens als Alternative zum Standard-8-K-Interpreter ein 12-KByte-Basic entwickelt, das auf Floppy oder in Form dreier EPROMs zur Verfügung steht, die in den drei freien Stecksockeln der Computerplatine Platz finden.

Hier eine Übersicht über die zusätzlich zum Standard-Basic vorhandenen Befehle:

INDEV — erlaubt das Setzen des Input-Flag auf ein beliebiges Eingabegerät (Tastatur, Kassette, User usw.).

OUTDEV — setzt das Output-Flag auf ein beliebiges Ausgabegerät.

JSR — gestattet es, Maschinensprache-Unterprogramme von Basic aus anzuspringen (wie SYS beim PET).

NORETURN — korrigiert den Stack, wenn man aus einem Unterprogramm ohne RETURN-Befehl zur nächsthöheren Programmebene springen möchte, z. B. zur Fehlerbehandlung.

CLEARSTACK — schließt alle offenen FOR-NEXT-Schleifen und beseitigt Unterprogramm-Rücksprungadressen auf dem Stack.

RENUMBER — erlaubt das Neunumerieren von Basic-Programm mit automatischer Korrektur aller Sprungadressen.

EDIT — wirkt wie der Change-Befehl des Editors, gestattet also das Ändern einzelner Zeichen in einer Programmzeile, ohne diese neu eingeben zu müssen.

QUIT — sorgt für das Verlassen des Basic-Interpreters und für den Rücksprung ins Monitorprogramm.

GODOS — führt einen Sprung zum Floppy-Disk-Betriebsprogramm durch.

PROGRAM — gestattet das Setzen der Basic-Zeiger auf einen beliebigen Adressenbereich, z. B. für ein im EPROM stehendes Basic-Programm.

MEMORY — ändert die Memory Size (von Basic belegter Arbeitsspeicher-Adressenbereich), ohne das vorhandene Programm zu zerstören.

RELOCATE — gestattet das Verschieben von Basic-Programmen in einen anderen Adressenbereich, z. b. um sie in EPROMs zu brennen.

OLOAD — erlaubt das Laden von Basic-Programmen von einem beliebigen Peripheriegerät.

XOR — Exklusiv-Oder

TRON — Trace-Modus einschalten (Einzelschrittbetrieb)

TROFF — Trace-Modus ausschalten

OKOFF — „OK"-Meldung ausschalten (für Display-Betrieb)

OKON — „OK"-Meldung einschalten

AUTO — Automatische Zeilennumerierung beim Eingeben von Basic-Programmen (beliebige Schrittweite). Abschalten mit CTRL C, Löschen einer Zeilennummer mit DEL.

GO — Programmstart ohne Variablenlöschung, solange die Programmlänge nicht verändert wurde (auch nach Error möglich!)

LISTERROR — zeigt mit einem Pfeil diejenige Stelle in einer Programmzeile an, wo sich genau der zuletzt aufgetretene Fehler befindet.

DSAVE — erlaubt das Abspeichern eines Basic-Programms auf Diskette.

DLOAD — lädt ein Basic-Programm von der Diskette.

GET — wirkt jetzt nicht nur auf der eingebauten Tastatur, sondern auch auf externe Eingabegeräte.

FORMSTR$ (A,B,C) — wirkt wie PRINT USING beim TRS-80 und liefert einen formatierten String von A; z. B. wird PRINT FORMSTR$(1,8,2) ausgeführt, indem 1.00 mit zwei Nachkommastellen gedruckt wird, wobei die niederwertigste Stelle an der achten Zeichenposition der Zeile steht.

HEXSTR$(A) — liefert die Zahl A als Hexadezimal-String.

BINSTR$(A) — liefert die Zahl A als binären String.

SAVE — gestattet das Abspeichern von Programmen auf ein beliebiges Ausgabegerät.

ATN — ist jetzt fest implementiert.

Der Basic-Arbeitsspeicher beginnt beim 12-K-Interpreter bei 0300 statt 0200, dadurch wurde ein Verlegen des Input-Puffers auf Page 2 möglich. Eingegebene Zeilen dürfen damit nun bis 234 Zeichen (statt bisher 60) umfassen; gleichzeitig wird der Zero-Page-Bereich A6...CC frei, solange man nicht EDIT und BINSTR$ verwendet.

Das ist aber noch nicht alles an Neuigkeiten des 12-K-Basic. Wenn man bisher nach INPUT sofort Return drückte, wurde das Programm abgebrochen. Das kann jetzt nicht mehr passieren. Ein Programmabbruch kann nur noch durch Shift-F2 herbeigeführt werden. Drückt man dann CTRL C, wird das Programm verlassen; drückt man DEL, so wird der Programmablauf verlangsamt weitergeführt; und Space sorgt nach Shift-F2 für das Weiterlaufen mit normaler Geschwindigkeit.

Das Abspeichern von Datenfiles ist mit PRINT möglich, das später LADEN mit INPUT; dabei muß nur vorher mit OUTDEV bzw. INDEV das externe Speichergerät festgelegt werden. Damit entfällt die sonst notwendige, in Heft 1/1980 abgedruckte File-Routine.

Adressen, Zeilennummern oder numerische Variablen dürfen nicht mehr nur dezimal, sondern durch Voranstellen des Dollar-Zeichens auch hexadezimal bzw. mit einem vorangestellten Prozent-Zeichen binär eingegeben werden, z. B. PRINT 245 + $B23A – %101001.

Die Möglichkeit, die Bereitschafts-Meldung „OK" abzuschalten, hat sich als notwendig erwiesen, wenn man mit dem einzeiligen Display arbeitet. Drückt man z. B. PRINT 2 + 3, so erscheint normalerweise nicht 5, sondern OK, weil das Ergebnis sofort überschrieben wird. Erst nach OKOFF arbeitet der Computer wieder wie von früher gewohnt. Fehler werden im vollen Klartext angezeigt.

Der 12-KByte-Basic-Interpreter übertrifft somit die meisten anderen heute lieferbaren Tischcomputer der Preisklasse unter 10 000 DM bei weitem. Zusammen mit einem externen Terminal sind AIM-65 bzw. PC-100 auch als Basic-Computer damit fast unschlagbar geworden. Fe.

```
10 GOSUB1200:PRINT:PRINTX$
20 GOTO10
1200 X$="":REM INPUT
1210 PRINTCHR$(230)"|";
1215 GETY$:IFY$=""GOTO1215
1220 Y=ASC(Y$):IFY=13ANDLEN(X$)>0THENPRINT" ":RETURN
1230 IFY$="|"ANDLEN(X$)>0THENX$=LEFT$(X$,LEN(X$)-1):PRINT"|  ||";:GOTO1210
1240 IFY<32ORY>127ANDY<160GOTO1215
1250 X$=X$+Y$:PRINTY$;:GOTO1210
```

Unterprogramm für die INPUT-Funktion (ab Zeile 1200) und kurzes Testprogramm (ab Zeile 10). Die senkrechten Striche nach PRINT-Anweisungen bedeuten „Cursor nach links"; z. B. steht in Zeile 1230 die Folge Cursor nach links, Space, zweimal Cursor nach links

Gefahrloser INPUT

Viele Tischcomputer verlassen das laufende Programm, wenn bei einer INPUT-Anweisung sofort Return gedrückt wird, ohne daß Daten eingegeben werden. Das kann zu Problemen führen, wenn ungeschultes Personal am Computer arbeitet, da der Programmabbruch meist zum Variablenverlust führt.

Das Unterprogramm im *Bild* dient als Ersatz für INPUT beim Einlesen von Strings und ist für den Computer CBM 3001 geschrieben (Business-Version, Großschreibung bei gedrückter Shift-Taste). Es läßt sich prinzipiell aber auch auf andere Computer übertragen.

Da die GET-Funktion zum Einlesen von Zeichen verwendet wird, kann das laufende Programm jetzt nur noch mit Break abgebrochen werden. Eine Besonderheit ist, daß mit der Taste „Cursor nach links" auch Korrekturen möglich sind. Die Übergabe der Zeichenkette erfolgt bei der Rückkehr aus dem Unterprogramm in der Variable X$. Als Cursor wurde hier ein nicht blinkendes, schraffiertes Feld gewählt (CHR$(230)). Selbstverständlich ist mit dieser INPUT-Funktion das Redigieren mit „Cursor up" und „Cursor down" nicht mehr möglich; dies ist aber ohnehin bei ungelerntem Bedienpersonal nicht zu empfehlen. Noch ein Vorteil der Routine: Die eingegebenen Strings dürfen bis zu 255 Zeichen lang sein, während sie beim normalen „Input" auf 80 Zeichen begrenzt sind. Fe.

Meine Meinung...

Computer müßte man sein!

In Schreiben an die Deutsche Amateur-Fernschreib-Gruppe (DAFG) und an den Deutschen Amateur-Radio-Club (DARC) gestattete das Bundespostministerium, wie bereits in Heft 24/1980 berichtet, den Funkamateuren bei Funkfernschreiben auch dann die Ausstrahlung ihres Rufzeichens im Morsecode, wenn sie eigentlich wegen ihrer Lizenzklase C nicht zum Morsen berechtigt sind. Einzige Voraussetzung: Das Rufzeichen darf nicht von Hand gemorst werden, sondern muß einem Automaten entstammen.

Nun ist es also tatsächlich soweit: Computer haben hochoffiziell mehr Rechte als ihre menschlichen Erbauer und Betreuer! Denn dem C-Funkamateur selbst ist es nach wie vor untersagt, selbst die Morsetaste zu betätigen. Abgesehen von der Bevorzugung von Automaten stellt dies wohl das merkwürdigste Ergebnis jener Diskussion dar, die auch Funkamateure untereinander über die zukünftige Bedeutung des Morsens bei der drahtlosen Nachrichtenübertragung abhielten.

Offenbar ist noch nicht jedermann klar, daß auch die Telegrafie trotz aller Maschinenstürmerei „nur" noch *eine* von mehreren Möglichkeiten digitaler Nachrichtenübermittlung ist; ein spezieller, wenn auch alter Code, der anderen hinsichtlich seiner Störanfälligkeit eher unterlegen ist. Konsequent wäre gewesen, bei Einführung der kürzlich herausgekommenen neuen Amateurfunk-Bestimmungen, das Morsen für Inhaber der C-(UKW-)Lizenz ebenso freizugeben wie zum Beispiel das Funkfernschreiben.

Oder ist in Wirklichkeit vielleicht Morsen mit C-Lizenz ohnehin schon freigegeben? Die Morse-Identifikation bei Funkfernschreiben wurde schließlich deshalb eingeführt, weil jetzt beliebige Fernschreibcodes verwendet werden dürfen – und auch Telegrafie ist für einen Computer ja nichts weiter als ein ganz normaler Digitalcode. Jedenfalls braucht der Besitzer eines Computers bei Morsetelegrafie nichts anderes zu tun, als wenn er im Baudot-Code fernschreibt – nämlich Tasten zu drükken... Herwig Feichtinger

µC-Nachrichten

Die Berliner Firma MCS (Micronic Computer Systeme GmbH) wird den 6502-Lerncomputer **Alpha-1** nicht mehr als Bausatz, sondern nur noch als Fertiggerät liefern. Ein kleines Trostpflaster: Im Preis ist nun auch der vollständige Dokumentationssatz enthalten. F'

In Heilbronn öffnete am 24. Oktober der **„Computer-Markt"** seine Pforten. Außer dem Hardware-Verkauf wird Analyse von Anwenderproblemen, Systemservice, Leasing, Schulung und Software-Entwicklung geboten. Zum Programm gehören Labels wie Apple, DAI, Tandy, Kiss (bzw. Alphatronic) und Kontron.

Die CPU-Story

Obwohl es heute einige Dutzend Typen von Mikroprozessoren gibt, lassen sie sich doch ganz grob in zwei „Sippen" einteilen – die 8080-Sippe und die 6800-Sippe. Sehen wir uns einmal an, wie es zu den beiden Gruppierungen kam und worin sie sich prinzipiell unterscheiden. Vielleicht dient dieser Beitrag damit ein wenig dazu, Scheuklappen bei manchen eingefleischten Befürwortern einer dieser CPU-Sippen, welcher auch immer, zu beseitigen.

Am Anfang war der „4004"

Erfunden wurde der Mikroprozessor 1969 von Dr. Ted Hoff bei Intel. Er baute 1970 den ersten 4-bit-Mikroprozessor, den 4004. Wenig später wurde dieser noch eher taschenrechnerorientierte Chip vom ersten 8-bit-Mikroprozessor gefolgt, dem 8008 – einer CPU in p-Kanal-MOS-Technologie mit sechs Arbeitsregistern und einem Stackregister für Unterprogramme. 1973 bautel Intel eine NMOS-Version des 8008, den 8080; der neue Typ besaß zusätzliche Befehle und kürzere Ausführungszeiten. Der 8080 war lange Zeit eine Art Industrie-Standard, und bis 1974 gab es praktisch keinen ernstzunehmenden Konkurrenten!

Dann aber kam Motorola mit einem Produkt auf den Markt, das sich sehr stark vom 8080-Konzept unterscheidet, der CPU 6800. Die Motorola-CPU hatte plötzlich keine besonderen Ein/Ausgabe-Befehle mehr, weil Motorola gleichzeitig Ein/Ausgabe-Bausteine herausbrachte, die sich wie normale Speicherzellen adressieren lassen (Memory mapped I/O). Damit ist es im Gegensatz zum 8080-Konzept möglich, beliebige Befehle (z. B. Addition oder Subtraktion) auch auf I/O-Bausteine anzuwenden. Zwar kann man an den 8080 auch memory-mapped-Bausteine anschließen, so daß dieser Nachteil entfällt; immerhin ist es aber das Verdienst von Motorola, dieses moderne Konzept überhaupt eingeführt zu haben.

Noch ein wesentlicher Punkt unterscheidet den 6800 vom 8080: Während der 8080 registerorientierte Befehle besitzt und wegen der vielen dafür erforderlichen CPU-internen Register eine relativ große Chipfläche benötigt, arbeitet der 6800 speicherorientiert. Spezielle Adressierungsarten, die der 8080 nicht kennt, erlauben es, 256 Bytes des externen Arbeitsspeichers (RAM) als Register zu verwenden, wofür im Programm nach dem Operationscode des Befehls nur ein Adreßbyte erforderlich ist.

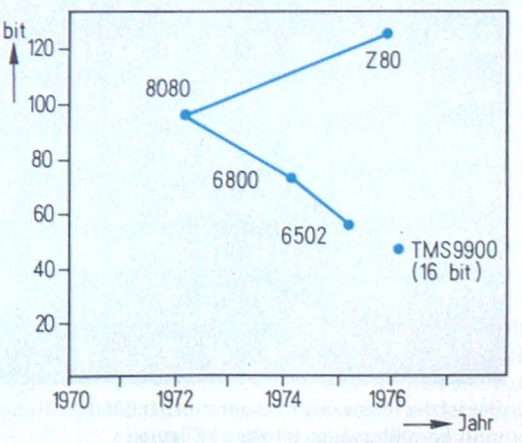
Dritte Generation: 6502 und Z-80

Der 6800 ähnelt damit schon mehr dem Konzept größerer Computer, konnte aber, da der 8080 immerhin vier Jahre früher existierte, dessen Verkaufserfolge nicht einholen.

8080 und 6800 waren die Grundsteine zweier Familien, die später schnell Zuwachs erhielten. Auf der 8080-Seite entwickelten 1976 ehemalige Intel-Angestellte den Z-80 von Zilog, der zum 8080 voll software-kompatibel und mit neuen Befehlen und Adressierungsarten verbessert ist; auch die Interrupt-Behandlung ist deutlich komfortabler als beim Vorläufer. Wegen der erzwungenen 8080-Kompatibilität und der Tatsache, daß der 8080 die meisten der 256 möglichen Bitkombinationen schon mit Operationscodes belegt hatte, mußte Zilog teilweise zu vier Byte langen Befehlen greifen. Zilogs Ziel war es, auf dem Erfolg des 8080 aufbauend, einen deutlich verbesserten Prozessor herzustellen, der ruhig etwas teurer sein durfte.

Einen anderen Weg ging 1975 MOS Technology, ein Unternehmen, das später von Commodore übernommen wurde, mit der CPU 6502. Ziel war die Herstellung eines möglichst preiswerten und dennoch leistungsfähigen Prozessors für den Einsatz in Massenprodukten. Zu diesem Zweck schufen ehemalige Motorola-Entwickler eine Art „abgemagerten" 6800: Um die Chipfläche zu verringern, ließ man eine Anzahl nur selten benötigter 6800-Befehle weg. Ebenso entfiel der zweite Akkumulator, und während der 6800 nur ein 16-bit-Indexregister besitzt, enthält der 6502 zwei Indexregister, die aber nur 8 bit lang sind. Aber der 6502 erhielt sieben Adressierungsarten mehr (insgesamt 13!). Ferner ist es möglich, seine Arithmetik zwischen binär und BCD hin- und herzuschalten. Der Erfolg stellte sich schnell ein: Schon 1978 war der 6502 weltweit der meistverkaufte Mikroprozessor und schnitt wegen seiner neuartigen Pipelining-Architektur auch bei Geschwindigkeits-Vergleichen oft als Bester ab.

Welcher der beiden Familien wird wohl die Zukunft gehören – der registerorientierten 8080-Familie mit speziellen I/O-Befehlen oder der speicherorientierten 6800-Familie mit memory mapped I/O? Die Antwort darauf geben die neuesten 16-bit-Prozessoren von Zilog und Motorola. Zilog's Z-8000 ist bereits deutlich memory-mapped-orientiert, während Motorola's 68000 einen recht umfangreichen Registersatz besitzt. Das Optimum liegt, wie so oft, in der Mitte.



Die Register-Speicherkapazität auf dem Prozessorchip ist kein Maßstab für den Programmierkomfort: Der „beste" Prozessor dieses Diagramms, die 16-bit-CPU TMS 9900, besitzt sogar am wenigsten Register auf dem Chip

Tischcomputer: Die CPU ist Nebensache

Welcher Prozessor ist nun der „Beste"? Für jeden individuellen Einsatzfall gibt es sicher einen Prozessor, dessen Eigenschaften ihn als besonders geeignet erscheinen lassen; allgemein läßt sich die Frage allerdings nicht beantworten. Nicht vergessen werden sollte, daß die Systemeigenschaften nur zu einem geringen Teil von der CPU selbst bestimmt werden, besonders seit es „intelligente" Peripheriebausteine mit programmierbaren Interrupt-Timern, Schieberegistern, Handshake-Steuerleitungen usw. gibt (z. B. 6522).

Deshalb sollte man sich beim Kauf eines Computers auch nicht nur daran orientieren, welche CPU verwendet wird. Viel wichtiger ist: Sind Interrupt-Timer eingebaut; sind I/O-Ports vorhanden, über die eine parallele Ein/Ausgabe stattfinden kann; ist eine serielle Schnittstelle (z. B. RS-232) für den Anschluß von Druckern vorhanden; ist ein IEC-Bus-Anschluß vorhanden, um auch Meßgeräte anschließen zu können; wie schnell und zuverlässig arbeiten externe Speichermedien (z. B. Kassette); entsprechen Tastatur und Display den Anforderungen – und so weiter. Und, nicht gerade unwichtig: Was kostet das System – und zwar nicht irgendeine abgemagerte Version ohne Schnittstellen, sondern das wirklich verwendbare System.

Ganz offensichtlich spielt der Mikroprozessor selbst also eine vergleichsweise geringe Rolle bei der Beurteilung eines Computers. Daraus sollte man den Schluß ziehen, nicht überflüssigerweise darüber zu diskutieren, ob Z-80 oder 6502 am besten für Tischcomputer geeignet sind.
(Nach einem Vortrag von Herwig Feichtinger auf dem FUNKSCHAU-Mikrocomputer-Seminar am 24.9.1980)

8080-Arithmetik-Routinen

Unter dem Titel „Arithmetik-Routinen für den 8080" veröffentlichten wir in dem jüngst erschienenen Sonderheft „Mikrocomputer-Anwendungen" ein umfassendes Programm, wie man Z-80- und 8080-Computern das Rechnen beibringen kann.

Leider ging dabei ein Teil des Listings verloren, nämlich der Teil von 4F50 bis 4FFF; er ist im *Bild* nachträglich wiedergegeben.

Der Subtraktionsalgorithmus ist in den Routinen für den Z-80 ausgelegt. Da die Programme aber auch für 8080-Anwender nützlich sind, sei erwähnt, daß wegen einer anderen Flag-Struktur dieser CPU eine kleine Änderung notwendig ist.

Zunächst ist das Carry zu setzen. HL zeigt auf einen Summanden bzw. den Minuenden und auf das Ergebnis, dessen MSD mit dem Anfang einer Page zusammenfallen muß.

Zunächst muß man durch Addition von 9A das Komplement zu 100 im Akku (über DE geladen) bilden. Nach DAA speichert man das Ergebnis via HL ab. Dann kann man die Pointer E und L jeweils um 1 dekrementieren und den Akku wieder über DE laden. Diese Schleife wird ausgeführt, bis Register L zu Null geworden ist. Rö/Fe

```
4F50  00>4C D1 D5 EB CD 1E>4C E1 01 08 00 09 E9 31 FD
4F60  43 3E 45 CD FA>4F CD 91 40 78 3C C0 41 C9 E5 2E
4F70  04 CD 00>4C 2D 36 00 C2 74>4F 2E 10 45 CD 40 0B
4F80  CA 7D>4F 77 2C 05 CA 5E 4F FE 0D C2 7D>4F 54 1E
4F90  10 0E 00 06 FF 2E 30 1A FE 2D C2 A6>4F C5 4D 1C
4FA0  2E 06 7E B1 77 C1 1A FE 45 CA D1>4F FE 0D CA DC
4FB0  >4F FE 2E C2 BA>4F CD 69>4F 1C 1A E6 0F 12 79 A7
4FC0   1F 6F 1A DA CA>4F 0F 0F 0F 0F 86 77 0C 1C C3 A6
4FD0  >4F CD 69>4F 1C 2E C0 0E 0E C3 97>4F CD 69>4F 2E
4FE0  06 7E B0 77 CD 7F>4C E1 1E 00 C3 17>4C 7E 07 07
4FF0  07 07 CD F6>4F 7E E6 0F C6 30 D7 00 00 C9 00 00_
```

Das ist der fehlende Programmteil der 8080-Arithmetik-Routinen. Das Byte bei 4FE5 muß korrekterweise 88 statt 7F lauten

Polar-koordinaten-Umrechnung in Basic

Das im *Bild* in einer PET/CBM-Version aufgelistete Programm gestattet die Umrechnung von Polar- in kartesische Koordinaten und umgekehrt in jedem beliebigen Winkelmodus. Das Programm erläutert sich während des Ablaufs von selbst, so daß auf eine detaillierte Bedienungsanleitung verzichtet wird. Thomas Prinz

```
  5 PRINT"POLARKOORDINATEN - UMRECHNUNG"
  6 PRINT:PRINT "WINKELMODUS EINGEBEN:"
  7 PRINT "A=(DEG 1,RAD 2,GON 3)":A=0
  8 GOSUB 110
  9 PRINT"P IN R )A( R IN P )B(
    OUT )O( STOP )S("
 10 FOR X=1 TO 20:NEXT :GET C$
 12 IF C$="A" THEN 50
 14 IF C$="B" THEN 80
 16 IF C$="O" THEN 100
 18 IF C$="S" THEN 30
 20 IF C$="W" THEN  6
 29 GOTO 10
 30 STOP
 50 PRINT:PRINT" P  IN  R  MODUS"A
 51 PRINT:INPUT "WINKEL - P";W,P
 52 IF A=1 THEN 55
 53 IF A=2 THEN 58
 54 IF A=3 THEN 62
 55 X = P*(SIN((W*π)/180))
 56 Y = P*(COS((W*π)/180))
 57 GOTO 65:REM DEG
 58 X = P*(SIN(W))
 59 Y = P*(COS(W))
 60 GOTO 65:REM RAD
 62 X = P*(SIN((W*π)/200))
 63 X = P*(COS((W*π)/200))
 65 PRINT"X"X,"Y"Y:PRINT"P"P,"MODUS"A,"W"W
 70 GOTO 9
 80 PRINT:PRINT" R  IN  P  MODUS"A
 81 PRINT:INPUT " X  -  Y";X,Y
 82 P=(((X↑2)+(X↑2))↑0.5)
 83 IF A=1 THEN 86
 84 IF A=2 THEN 88
 85 IF A=3 THEN 90
 86 W =  (ATN(X/Y)* (180/π))
 87 GOTO 95
 88 W =  (ATN(X/Y))
 89 GOTO 95
 90 W =  (ATN(X/Y)* (200/π))
 95 PRINT"X"X,"Y"Y:PRINT"P"P,"MODUS"A,"W"W
 99 GOTO 9
100 FOR Y=1 TO 20:PRINT:NEXT
102 FOR F=1 TO 20:GET X$
104 IF X$= "R" THEN 10
105 NEXT : GOTO 100
110 REM GET FUER A
111 PRINT:PRINT"MODUS FUER A:"
112 FOR X=1 TO 50:NEXT :GET A
113 IF A=1  THEN 9
114 IF A=2  THEN 9
115 IF A=3  THEN 9
116 GOTO 112
```

Was für manche programmierbare Taschenrechner kein Problem ist, macht bei Basic-Computern dieses Programm erforderlich

Dr. Erhard Kisker

Höhere Auflösung bei PET-Grafiken

Eine sehr nützliche Funktion, die es z. B. beim TRS-80 gibt, fehlt im Befehlsvorrat des PET: die „Set"-Funktion, die es gestattet, an einer durch zwei Parameter X und Y bestimmten Stelle des Bildschirmes einen Punkt aufleuchten zu lassen. Dies ist besonders nützlich zum Darstellen mathematischer Funktionen und von Histogrammen, wird aber auch in vielen Spielen benutzt.

Ein Programm, im Maschinencode geschrieben, das eine Auflösung von 50 x 80 = 4000 Punkten zuläßt, ist im folgenden beschrieben. Diese hohe Auflösung wird mit Hilfe eines Tricks erreicht. Das zur Verfügung stehende Bildschirmraster besteht nämlich unabänderlich, durch das PET-Video-Interface festgelegt, aus 25 x 40 = 1000 Punkten. Eine bessere Auflösung ist nur dadurch möglich, die zu zeichnende Kurve mit Hilfe spezieller Zeichen darzustellen, die noch innerhalb der einzelnen Charaktere eine der Kurve angepaßte Struktur haben. Als für diesen Zweck generell brauchbarer Zeichensatz bieten sich die in *Bild 1* abgebildeten Zeichen an, die noch innerhalb des Platzes eines Zeichens eine Auflösung von 2 x 2 ermöglichen. Damit erhält man insgesamt die Auflösung von 2 x 2 x 25 x 40 = 4000 Punkten.

Das Plotten mit Hilfe dieser Zeichen ist nun aber nicht ganz einfach, denn es muß jeweils aus 16 Zeichen das richtige herausgesucht werden. Vor dem Setzen eines neuen „Mini"-Punktes muß nämlich geprüft werden, ob an der betreffenden Bildschirmposition (die im 25 x 40-Raster bestimmt ist) bereits eines der Zeichen aus Bild 1 gesetzt ist. In diesem Falle besteht dann das „Setzen" eines Punktes in dem Ersetzen des vorhandenen Zeichens durch dasjenige Zeichen, das der Kombination des vorhandenen Zeichens mit dem neu zu setzenden Punkt entspricht. Diese Prozedur wird zweckmäßigerweise vom Computer selbst durchgeführt. Dies geschah ursprünglich mit Hilfe eines Basic-Programmes, welches jedoch wegen der Vielzahl von notwendigen Entscheidungen sehr langsam ist. Außerdem haben solche Basic-Routinen den Nachteil, daß man sie nur umständlich vom Band her an vorhandene Hauptprogramme nachträglich anfügen kann.

Daher wurde das Programm in den Maschinencode umgeschrieben. Es wurde nach und nach so kompromiert, daß es in Pufferspeicher für den zweiten Kassettenrecorder (033A...03F9) hineinpaßt. Dies ist sehr praktisch, da somit kein Basic-Speicherplatz benötigt wird und ein Überschreiben durch Basic-Programme nicht erfolgen kann. Der zweite Kassettenrecorder kann dann natürlich nicht benutzt werden.

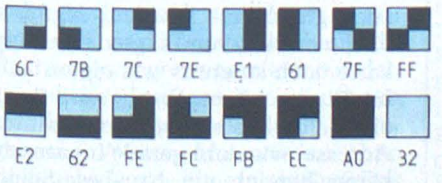


Bild 1. Die höhere Auflösung des PET-Bildschirms wird durch Ausnutzung von Grafiksymbolen erreicht

```
1 GOTO1000
5 PRINT"♥
10 M=0:L=49:K=79:X=42
20 FORI=MTOL:POKEX,40:A=USR(I):NEXT
40 FORI=M0TOK:POKEX,I:A=USR(25):NEXT
60 FORI=MTOK:POKEX,I:A=USR(SIN(I/8.5+PH)
   *25+25):NEXT
100 T=TI
120 IFTI-T<400THEN120
200 PRINT"♥JETZT EINGABE VON X,Y-PAAREN
220 PRINT"0<=X<=79,0<=Y<=49
230 FORK=0TO4
240 INPUT"X,Y";X,Y:POKE42,X:A=USR(Y)
260 NEXTK
300 PRINT"FERTIG":END
1000 DATA 32,108,123,124,126,225,255
1010 DATA 98,226,127,97,254,252,251
1020 DATA 236,160,126,127,97,226,126
1030 DATA 251,236,252,226,127,97,160
1040 DATA 252,251,236,160,124,225,255
1050 DATA 124,226,225,255,254,226,251
1060 DATA 236,254,160,251,236,160,123
1070 DATA 98,123,255,97,254,255,98
1080 DATA 236,252,97,254,252,160,236
1090 DATA 160,108,108,98,225,127,225
1100 DATA 254,98,251,127,252,254,252
1110 DATA 251,160,160,169,0,133,41
1120 DATA 32,167,208,169,50,229,180
1130 DATA 133,180,70,179,106,102,41
1140 DATA 10,10,10,72,10,38,179
1150 DATA 10,38,179,133,180,24,104
1160 DATA 101,180,133,180,144,2,230
1170 DATA 179,165,42,24,74,133,42
1180 DATA 168,102,41,24,169,128,101
1190 DATA 179,166,180,133,180,134,179
1200 DATA 177,179,160,16,217,57,3
1210 DATA 240,5,136,208,248,160,1
1220 DATA 132,34,36,41,48,10,80
1230 DATA 4,169,48,208,12,169,16
1240 DATA 208,8,80,4,169,64,208
1250 DATA 2,169,32,24,101,34,168
1260 DATA 185,57,3,164,42,145,179
1270 DATA 96,70,71
1280 FORI=826TO1015:READA:POKEI,A:NEXT
1290 POKE1,138:POKE2,3
2000 GOTO5
```

Bild 2. Das Basic-Programm generiert ein Maschinenprogramm im Kassettenbuffer 2 und ist für das „alte" Betriebssystem des PET 2001 ausgelegt

Der Befehlsatz des 6502-Mikroprozessors eignet sich vorzüglich für die gestellte Aufgabe, so daß die Ausführungszeit sehr kurz ist. Es dauert nur ca. 7 ms, bis ein Punkt gesetzt ist. Damit ist diese Funktion genauso schnell wie die eingebaute Set-Funktion des TRS-80. Als Eingabewerte X,Y werden Werte zwischen 0 und 49 (Y) sowie zwischen 0 und 79 (X) zugelassen. Der X-Wert wird mit Hilfe der POKE-Funktion in die Zelle 2A transferiert. Dann wird die USR-Funktion aufgerufen, wobei Y das Argument ist. Nach dem USR(Y)-Aufruf befindet sich das Argument im Fließkommaakkumulator, der sich beim PET in den Speicherzellen B0–B5 befindet. Dann springt der PET zur Subroutine, deren Adresse in den Speicherzellen 0001 und 0002 angegeben ist. Dorthin ist *vor* dem ersten Aufruf

der USR-Funktion die Adresse 038A mit Hilfe von Poke-Befehlen zu schreiben. Dort beginnt nämlich die Routine, die aus den Werten X und Y sowie aus dem sich an der Bildschirmstelle X/2, Y/2 befindlichen Zeichen den richtigen Charakter heraussucht. Dazu sind dem Programm fünf Tabellen vorangestellt; jede ist 16 Byte lang und besteht aus den Zeichen von Bild 1. Aus der ersten Tabelle wird das schon gesetzte Zeichen herausgesucht. Je nachdem, ob X und Y gerade oder ungerade waren, wird dann aus einer der vier folgenden Tabellen das Zeichen genommen, das der Kombination des vorhandenen und des neu zu setzenden Zeichens entspricht. Die Tabellen sind ab Adresse 033AH vor das eigentliche Programm gesetzt.

Leider sind bei der verbesserten Version des PET, dem CBM, die Bedeutungen der „Page-Zero"-Adressen anders als beim PET. Auch haben fast alle Basic-Subroutinen andere Adressen, so auch FLPINT. Daher müßten diese Adressen für die Benutzung auf dem CBM abgeändert werden. Hier sei auf die ROM-/RAM-Tabellen im FUNKSCHAU-Sonderheft „Mikrocomputer-Anwendungen" verwiesen.

Speichertest per Zufall

Das Prüfen eines Schreib-Lese-Speichers (RAM) ist ein interessantes Problem der Kombinatorik. Es genügt nämlich nicht, einfach alle Speicherzellen zuerst mit lauter Nullen und dann mit lauter Einsen zu beschreiben, da dann z. B. eine Duplizierung von Adressenbereichen durch kurzgeschlossene Leitungen im Adressenbus o. ä. nicht erkannt werden könnte.

Oft genug wurde schon überlegt, welche Zahlenfolgen am besten geeignet wären, bei allen denkbaren Speicherfehlern eindeutige Ergebnisse zu liefern. Eine der sichersten Methoden ist dabei die Verwendung einer Zufallszahlenfolge, da dann die Wahrscheinlichkeit für eine verhängnisvolle Korrelation zwischen Testzahlen und fehlerhaften Speicheradressen äußerst klein ist.

Das im *Bild* aufgelistete 6502-Programm mit der Startadresse 0000 benutzt ein von Jim Butterfield entwickeltes Zufallszahlen-Programm. Die Startadresse des fraglichen RAM-Bereiches ist in 00E0 und 00E1 (L, H), die Endadresse in 00E2 und 00E3 abzulegen. (Wenn z. B. 0400 als Endadresse gewählt wird, ist 03FF die letzte getestete RAM-Adresse.) Wenn nach dem Testprogramm-Durchlauf die Speicherzellen 00E4 und 00E5 Null sind, so trat kein Fehler auf. Es empfiehlt sich, den Vorgang mehrmals zu wiederholen, um jede Speicherzelle mit unterschiedlichen Bitkombinationen zu testen.

Das Programm schreibt in alle Zellen des zu testenden Speichers 8-bit-Zufallszahlen und errechnet gleichzeitig eine 16-bit-Prüfsumme dieser Zahlen. Dann wird der Speicherbereich ausgelesen und mit Hilfe der Prüfsumme auf Veränderungen untersucht. Die Laufzeit beträgt etwa eine Sekunde für 6 KByte. Natürlich ist es nicht möglich, die Zero-Page zu testen, da sie vom Programm belegt wird. Außerdem treten Fehler auf, wenn man versucht, den benutzten Stack-Bereich (01FE...01FF) zu testen, da die Zufallszahlen dort von Rücksprungadressen überschrieben werden.

Der Sprungbefehl an der Adresse 003E führt zum Monitorprogramm des hier verwendeten Mikrocomputers KIM-1. Beim AIM-65 ist er auf 4C 82 E1 zu ändern. Der Erfindungsgabe des Lesers sei es überlassen, das Testprogramm mit benutzerfreundlichen Ein- und Ausgaberoutinen zu versehen und zu einer Schleife für mehrere Durchläufe zu erweitern. Fe.

```
0000 A0 00     LDY #00
0002 84 E4     STY E4
0004 84 E5     STY E5
0006 20 41 00  JSR 0041
0009 38        SEC
000A A5 E9     LDA E9
000C 65 EC     ADC EC
000E 65 ED     ADC ED
0010 85 E8     STA E8
0012 A2 04     LDX #04
0014 B5 E8     LDA E8,X
0016 95 E9     STA E9,X
0018 CA        DEX
0019 10 F9     BPL 0014
001B 91 E6     STA (E6),Y
001D 18        CLC
001E 65 E4     ADC E4
0020 90 02     BCC 0024
0022 E6 E5     INC E5
0024 85 E4     STA E4
0026 20 4A 00  JSR 004A
0029 90 DE     BCC 0009
002B 20 41 00  JSR 0041
002E A5 E4     LDA E4
0030 38        SEC
0031 F1 E6     SBC (E6),Y
0033 B0 02     BCS 0037
0035 C6 E5     DEC E5
0037 20 4A 00  JSR 004A
003A 90 F4     BCC 0030
003C 85 E4     STA E4
003E 4C 4F 1C  JMP 1C4F
0041 A6 E0     LDX E0
0043 86 E6     STX E6
0045 A6 E1     LDX E1
0047 86 E7     STX E7
0049 60        RTS
004A E6 E6     INC E6
004C D0 02     BNE 0050
004E E6 E7     INC E7
0050 A6 E7     LDX E7
0052 E4 E3     CPX E3
0054 90 04     BCC 005A
0056 A6 E6     LDX E6
0058 E4 E2     CPX E2
005A 60        RTS
```

Dieses 6502-Programm testet einen RAM-Bereich durch Einschreiben einer Zufallszahlen-Folge

Apple mit 64 KByte RAM

Wenn ein Mikrocomputer-System zum Zugriff auf die Zellen seines Schreib-Lesespeichers und seines Festwertspeichers einen 16 Bit breiten Adreßbus benutzt, dann kann es im günstigsten Fall auf 65 536 (= 2^{16}) Adressen zugreifen. Dies bedeutet, daß ein solches System von Haus aus nur mit soviel RAM-Zellen ausgerüstet werden kann, wie an Adreßbereich verbleibt, wenn man von dieser Zahl den von ROMs und Eingabe-Ausgabe-Bausteinen in Anspruch genommenen Adreßraum subtrahiert. Für Maschinen vom Typ Apple-II und ITT-2020 gilt diese Einschränkung ab sofort nicht mehr. Eine neue Zusatzkarte rüstet diese Rechner auf 64 K RAM auf und bietet darüber hinaus die Möglichkeit, durch einfache Software-Kommandos zwischen zwei residenten Betriebssystemen hin- und herzuschalten.

Schaltungstechnisch wird das durch einen Hardware-Trick erreicht, der in der mittleren Datentechnik und im Großrechnerbau in abgewandelter Form schon seit Jahren Stand der Technik ist: Ein und dieselbe Adresse gilt mehreren parallelgeschalteten Bauteilen, von denen aber nur jeweils eines aktiviert ist; jenes nämlich, das zu einem zuvor durch Software-Kommando aktivierten Block gehört. So ist es im Falle Apple-Plus zum Beispiel möglich, im Bereich hex D000 bis DFFF wahlweise das Betriebssystem-ROM oder einen von zwei RAM-Blöcken zu je vier KByte zu erreichen. Mit anderen Worten: 4096 Einzeladressen erlauben den Zugriff auf 12 288 Byte.

Wenn schon Umschaltung – dann auch gleich mit Komfort, sagten sich die Entwickler, und fügten ihrer Superkarte noch so etwas wie eigenständige Intelligenz hinzu: Ergeht zum Beispiel ein Schreibbefehl an eine bestimmte Adresse, wiewohl gerade im umschaltbaren Bereich ein Nur-Lese-Speicher aktiviert ist, so schließt das kartenin-

terne Mini-Betriebssystem daraus, der Programmierer habe das Umschalten vergessen, und holt das Versäumte für ihn nach. Komfortabel – aber auch gefährlich! Ein Programmierfehler könnte dazu führen, daß in einem solchen Fall das System in unheilbare Bewußtseinsspaltung verfällt. Aber auch daran dachten die Apple-Väter: Ein weiteres Software-Kommando dient zur Verriegelung („Write Protect", Schutz vor Neubeschreibung) einmal geladener RAMs.

Wozu nun all das? Zwar hätte diese Art von Erweiterung des Adressenbereichs allein den Aufwand schon gelohnt, bei der Verarbeitung großer Datenmengen geht's ja in aller Regel um so fixer, je weniger Daten auf externe Massenspeicher wie Floppyscheiben oder Hartplatten ausgelagert werden müssen; aber die Leute bei Apple im kalifornischen Cupertino geben ihrer „Language Card" als Zugabe ein komplettes Pascal-Betriebssystem mit auf den Weg, das von vier mitgelieferten Disketten schnell in die Karte geladen werden kann. Daher auch der Name „Language Card", Sprachenkarte: Die Hauptanwendung liegt in der Möglichkeit, künftig mit Hilfe dieser Erweiterungskarte neben PASCAL weitere Programmiersprachen in die Apple-II-Maschinen „Standard" und „Plus" laden zu können.

Erprobt hat der Autor die vom deutschen Apple-Generaldistributor Basis Münster vertriebene Language Card auf einem Apple-II-Plus mit Hilfe der Pascal-Version System II.1 der Universität San Diego. Rechner, Karte und PASCAL-System benahmen sich in mehrwöchiger Erprobung einwandfrei, lediglich die beiden Apple-Diskettenstationen, fabrikfrisch angeliefert und im schlichten Monitor- oder Basicbetrieb ohne Fehl und Tadel, zeigten bei der Zusammenarbeit mit Language Card und PASCAL gelegentliche Betriebsstörungen, die erst nach einem Neuabgleich der Laufwerkgeschwindigkeiten verschwanden. Was blieb, war eine erhöhte Empfindlichkeit gegenüber Zentrierfehlern, bisweilen muß die Magnetscheibe mehrfach in die Station eingeführt werden, bis sie richtig sitzt und vom System akzeptiert wird. Ursache dieses etwas übelnehmerischen Betragens der Laufwerke: Language Card und PASCAL schreiben dichter und mehr auf die Scheiben, als dies das Standard-Disketten-Betriebssystem DOS 3.2 des Apple gemeinhin tut. J.V.

AIM-65 mit Videoausgabe

In Heft 25/1979 der FUNKSCHAU war nachzulesen, wie man ein Video-Interface mit serieller Schnittstelle an den Mikrocomputer AIM-65 (bzw. PC-100) anschließt. Da jedoch z. B. von der Mainzer Firma Eltec sehr preiswerte Video-Interface-Platinen mit paralleler ASCII-Schnittstelle und positivem Strobe-Eingang zu haben sind (VIC-1), veröffentlichen wir nachstehend eine Methode, solche Platinen am AIM zu verwenden.

Die notwendige Software (*Bild*) ist minimal. Mit der Taste F1 wird die Video-Ausgabe initialisiert; dabei wird der DILINK-Vektor des AIM auf 0FD0 gesetzt, also auf die Routine zur Ausgabe eines Zeichens auf den Bildschirm. Außerdem wird Port A als Ausgang geschaltet.

Die Ausgaberoutine selbst gibt ein ASCII-Zeichen an Port A aus und hält den Strobe-Eingang des Interface auf „High", bis das nächste Zeichen eintrifft. Die Verzögerungszeiten bei den Steuerzeichen entsprechen den vom Hersteller der Interfacekarte VIC-1 angegebenen Werten und werden ohne Zuhilfenahme von Timern erzeugt. Das ASCII-Zeichen „Return" (hex 0D) wird automatisch durch „Line Feed" (hex 0A) ergänzt.

Die Ausgabe erfolgt grundsätzlich gleichzeitig auf dem Bildschirm und auf dem AIM-Display, da der Display-Buffer im RAM von Teilen des Monitorprogramms abgefragt wird. Schaltet man den AIM auf TTY-Betrieb, so wird die Videoausgabe unterbunden (auch die AIM-Tastatur ist dann abgeschaltet).

Ein Haken bei der Sache sei nicht verschwiegen: Die Taste „Delete", die der Korrektur falscher Eingaben dient, funktioniert zwar auf dem parallel mitlaufenden Display, nicht aber auf dem Bildschirm. Eine Lösung dieses Problems wird in einem späteren Heft vorgestellt werden; sie bedient sich der Abfrage des Display-Zeigers.

Apropos Basic: Beim Einstieg in den Basic-Interpreter mit Taste 5 muß vorher bereits F1 gedrückt worden sein. Bei „Memory Size" ist 4032 einzugeben (=hex FC0). Die *Tabelle* zeigt schließlich die notwendigen Verbindungen zwischen AIM-65 (Stecker J-1) und Videokarte.

Wer Schwierigkeiten hat, beim VIC-1 den letzten Buchstaben einer Zeile noch auf dem Bildschirm zu sehen, weil die Bildbreite sich nicht klein genug einstellen läßt, sollte auf der Interface-Platine die Leitung zu Pin 11 des 74132 auftrennen und statt dessen zu Pin 3/4/5 legen. Durch Vergrößern des 20-pF-Kondensators kann man die jetzt meist zu geringe Bildbreite auf den Sollwert bringen. Als Fernsehgerät sollte man übrigens unbedingt (!!!) ein mit Videoeingang versehenes, galvanisch vom Netz getrenntes Gerät verwenden. Fe.

```
<K>*=FC0
/29
 0FC0 A9 LDA #FF
 0FC2 8D STA A003
 0FC5 A9 LDA #D0
 0FC7 8D STA A406
 0FCA A9 LDA #0F
 0FCC 8D STA A407
 0FCF 60 RTS
 0FD0 48 PHA
 0FD1 20 JSR EB9E
 0FD4 29 AND #7F
 0FD6 8D STA A001
 0FD9 09 ORA #80
 0FDB 8D STA A001
 0FDE C9 CMP #A0
 0FE0 10 BPL 0FF6
 0FE2 A2 LDX #08
 0FE4 C9 CMP #8C
 0FE6 D0 BNE 0FEA
 0FE8 A2 LDX #A0
 0FEA A0 LDY #E0
 0FEC 88 DEY
 0FED D0 BNE 0FEC
 0FEF CA DEX
 0FF0 D0 BNE 0FEA
 0FF2 C9 CMP #8D
 0FF4 F0 BEQ 0FF9
 0FF6 4C JMP EEEC
 0FF9 A9 LDA #0A
 0FFB D0 BNE 0FD6

<K>*=10C
/
 010C 4C JMP 0FC0
```

Software zum Ansteuern eines Video-Interface mit paralleler Schnittstelle. Die Initialisierung erfolgt vom Monitorprogramm aus mit der Taste F1

Tabelle der Verbindungen

AIM-65 Port		*Interface VIC-1* Leitg.	
	Stecker		Stecker
PA0	14	D0	19c
PA1	4	D1	18c
PA2	3	D2	17c
PA3	2	D3	16c
PA4	5	D4	15c
PA5	6	D5	14c
PA6	7	D6	13c
PA7	8	Strobe	12c
GND	1	GND	1c
5 V	A	5 V	31c

Tandy's Farbcomputer

Tandy/Radio Shack hat außer dem bekannten TRS-80 nun drei weitere Computer im Programm: den von Sharp gefertigten Basic-Taschencomputer PC-1211, das schon erhältliche TRS-80-Modell II und schließlich den „Color Computer", der sich an einen NTSC-Farbfernsehempfänger anschließen läßt und, weil er nicht über einen eingebauten Bildschirm verfügt, noch peiswerter als der TRS-80 ist.

Trotz des geringen Preises verfügt der „Color Computer" über eine Reihe interessanter Features. Er arbeitet mit der CPU 6809 von Motorola, einer 8-bit-CPU mit interner 16-bit-Struktur und Hardware-Multiplikationsbefehlen – sicher das Modernste, was es heute auf dem 8-bit-Sektor gibt. Im Gegensatz zum TRS-80 ist auch eine serielle RS-232-Schnittstelle eingebaut, z. B. um ein Telefon-Modem anschließen zu können. Was im US-Preis von 400 Dollar allerdings nicht enthalten ist, das ist ein Kassettenrecorder – den muß man sich zusätzlich besorgen, aber dafür genügt ein Einfachst-Audio-Recorder für 50 DM. Der „Color Computer" enthält einen 6-K-Microsoft-Interpreter für Basic; es wird noch herauszufinden sein, ob es sich dabei um ein „echtes" 6809-Basic handelt oder um ein (mit dem 6809 kompatibles) Basic für den etwas älteren Prozessor 6800. Schließlich arbeitet der TRS-80 mit seiner CPU Z-80 auch „nur" mit einem 8080-Basic...

Das verwendete Kassetten-Aufzeichnungsformat ist wesentlich zuverlässiger und auch schneller als das des TRS-80. Im Originalzustand besitzt der 400-Dollar-Computer 4 KByte RAM, das sich einfach auf 16 KByte erweitern läßt; im Prinzip ist eine Erweiterung bis maximal 32 KByte möglich. Auf dem Bildschirm lassen sich Grafiken farbig mit einer Auflösung von 128 x 192 Punkten darstellen; allerdings werden allein dafür schon 6 KByte RAM benötigt. Im gleichen Speicherplatz bringt man auch eine Schwarzweiß-Grafik mit 256 x 192 Punkten unter.

Da das System insgesamt nur 8 K ROM enthält, von denen ein Teil ja schon für das Maschinensprache-Steuerprogramm (Bildschirmsteuerung, Tastaturabfrage usw.) benötigt wird, unterliegt der Basic-Interpreter einigen Einschränkungen. Zwar verfügt er über volle Fließkomma-Arithmetik, kennt aber keinerlei logarithmische Funktionen, und an Trigonometrie ist ihm nur SIN zuzumuten. Das ist ja nicht weiter schlimm, werden Basic-Spezialisten sagen, dann definiere ich mir eben neue Funktionen mit DEFFN! Leider hat das einen Haken: Auch die DEFFN-Anweisung ist dem „Color Computer" nämlich unbekannt.

Auf dem Bildschirm lassen sich 16 Zeilen mit je 32 Zeichen darstellen – leider keine Kleinbuchstaben, denn der Zeichengenerator umfaßt nur 64 „Characters". Zusätzlich zur hochauflösenden Grafik, die nur von Maschinensprache, nicht aber von Basic aus zugänglich ist, steht die „Low resolution graphics" zur Verfügung, die dem TRS-80 entspricht und ein Zeichenfeld in 2 x 2 Teilquadrate einteilt, so daß sich 32 x 64 Punkte ergeben. Diese Grafik ist auch ohne Speicherweiterung verwendbar.

Fazit: Der „Color Computer" ist preiswert und intelligent konstruiert – aber für den professionellen Anwender wohl nicht unbedingt ideal. Fe.

Literatur

Baker, Al: The TRS-80 Color Computer. Interface Age, 11/1980.

Umschaltbares Betriebssystem beim PET

Der Traum eines jeden PET-Benutzers ist nach der Anschaffung eines Druckers der Kauf einer Floppy. Dazu ist es erforderlich, den PET mit dem neuen Betriebssystem zu modifizieren. Nach dem Umrüsten laufen aber die vielen mühsam geschriebenen Basic- und Maschinenprogramme nicht mehr.

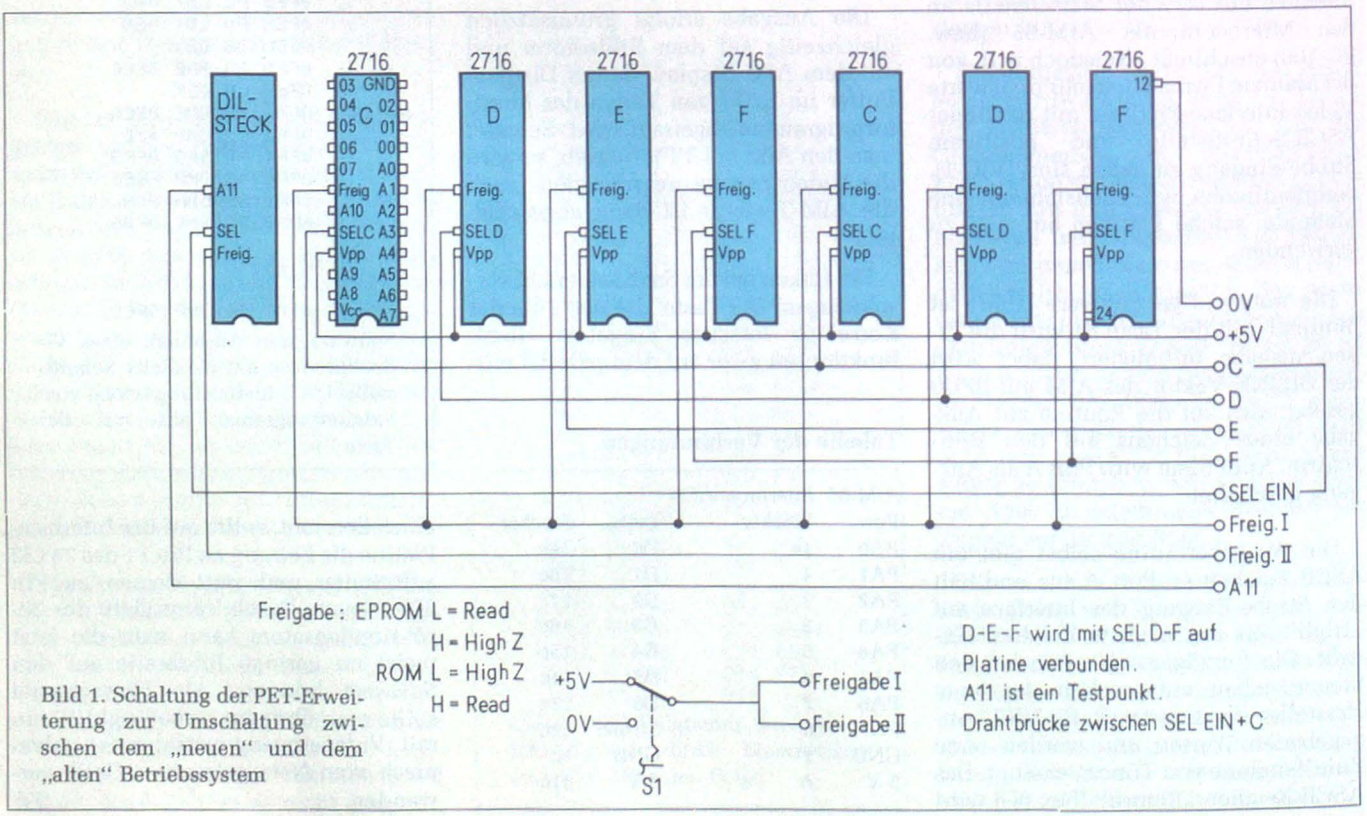


Bild 1. Schaltung der PET-Erweiterung zur Umschaltung zwischen dem „neuen" und dem „alten" Betriebssystem

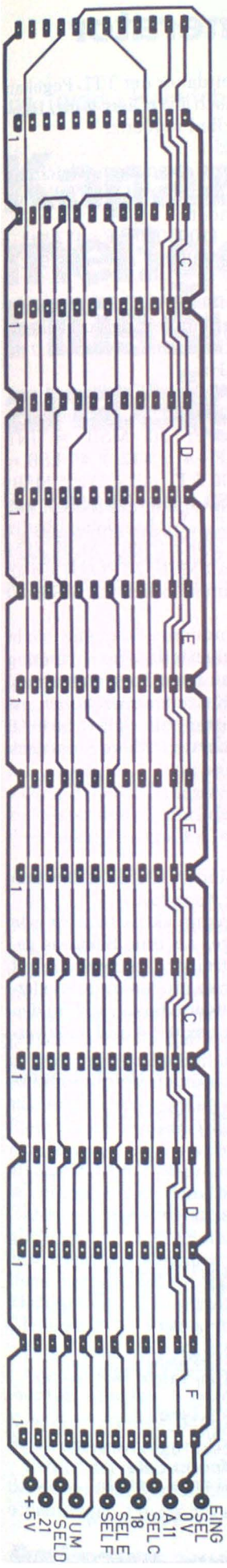


Bild 2.
Platinenoberseite der PET-Erweiterung

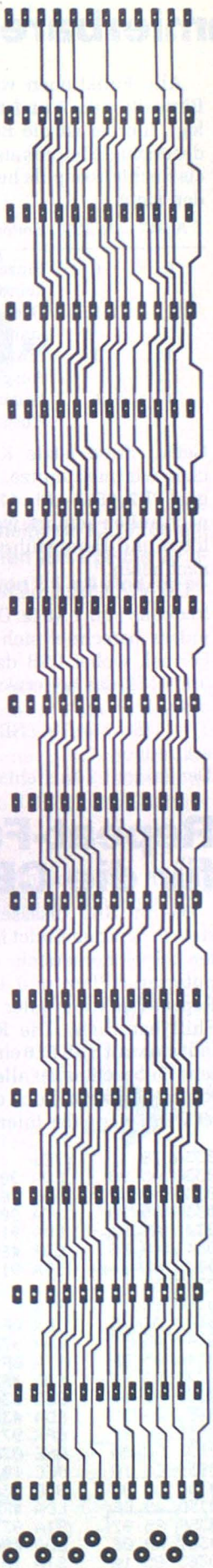


Bild 3.
Platinenrückseite. Die Bestückung geht unmittelbar aus der Lötaugen-Lage hervor: Benötigt werden sieben 2716-EPROMs

Das Umschreiben speziell der Maschinenprogramme ist meist zeitintensiver als eine Neuentwicklung. Abhilfe schafft ein umschaltbares Betriebssystem.

Bild 1 zeigt die Schaltung einer Erweiterung, die innerhalb des PET-Gehäuses Platz findet und über eine 24polige Leitung mit der Mutterplatine verbunden wird. Da die vier ROMs mit dem neuen Betriebssystem teuer und schwer zu beschaffen sind, werden entsprechende EPROMs eingesetzt (dazu ist es nötig, jemanden zu kennen, der das neue Betriebssystem auf EPROMs kopieren kann). Die Erweiterungsplatine in den *Bildern 2* und *3* bietet deshalb Platz für sieben EPROMs 2716 (beziehbar von den Autoren, Ahornstr. 9, 4174 Issum).

Willi Billen,
Willi Gietmann

Meine Meinung...

Der erlogene Mikroprozessor

Nicht selten findet man in Anzeigen und Prospekten Schlagworte wie „mikroprozessorgesteuert", „mit modernsten Mikroprozessoren aufgebaut" und ähnliches.

Was hier mit dem Wort Mikroprozessor gemacht wird, ist eine manchmal vielleicht sogar beabsichtigte Irreführung des Käufers von HiFi-Geräten, Fernsehempfängern, Schaltuhren, Anrufbeantwortern und anderen Geräten. Mikroprozessoren sind definitionsgemäß die zentralen Steuereinheiten in Mikrocomputern, und diese sind wiederum programmgesteuerte Systeme, die bestimmte Problemstellungen in serieller und nicht etwa, wie bei TTL-, CMOS- oder MOS-Logiken, in paralleler Arbeitsweise bewältigen.

Eine einfache CMOS-Logik als „Mikroprozessorsteuerung" zu bezeichnen, ist vielleicht verkaufswirksam, aber nun einmal schlicht erlogen. Man sollte ruhig die Seriosität von Herstellern unterschiedlichster Geräte daran messen, ob sie es nötig haben, mit Begriffen zu werben, die irreführend und technisch völlig falsch sind.

Herwig Feichtinger

PET als programmierbarer Frequenzgenerator

Trotz der immer besser werdenden Benutzerhandbücher hat der Mikrocomputer PET noch eine ganze Reihe versteckter Möglichkeiten, die den meisten Benutzern unbekannt sind. So werden insbesondere spezielle Hardwareeigenschaften nur durch Herumprobieren oder durch gewissenhaftes Studium der Original-Datenblätter herausgefunden. Wer hat diese aber schon; außerdem gehören hierzu doch eine Portion Erfahrung und gute Englischkenntnisse. So kommt es, daß unter PET-Besitzern so etwas wie ein schwarzer Markt für Tips, Tricks und besondere Speicheradressen existiert.

Dieser Beitrag soll zeigen wie ohne zusätzlichen Hardwareaufwand und mit relativ geringer, einfacher Software ein universell programmierbarer, sehr genauer Frequenzgenerator für TTL-Rechteckimpulse programmiert werden kann.

Zu verdanken ist dies dem sehr leistungsfähigen Baustein VIA 6522 (siehe FUNKSCHAU-Sonderheft „Hobbycomputer 2"). Dieser vielseitige Baustein hat insgesamt 16 Register: 2 bidirektionale 8-Bit Ein-/Ausgabe-Ports ORA, ORB, wovon der erste als User-Bus nach außen geführt ist, der zweite durch interne und externe Steuersignale belegt ist.

Zu jedem dieser Datenports existiert ein Datenrichtungsregister, in dem jedes einzelne Bit der Datenports als Ein- oder Ausgabesignal programmiert werden kann. Des weiteren gibt es zwei 16-bit-Timer, die zahlreiche Funktionen ausüben können wie z. B. Ereigniszählen, Impulsbreitenmessung und Erzeugung von einmaligen oder stetigen Rechteckimpulsen auf einem Ausgabekanal. Diese Anwendung soll uns im folgenden interessieren.

Zu erwähnen sind noch die Hilfsregister ACR und PCR, die eine Reihe von Kontroll- und Steuerfunktionen übernehmen sowie die Register IFR und IER, die bei der Interruptverarbeitung eine Rolle spielen. Hierauf soll an dieser Stelle nicht weiter eingegangen werden.

Die beiden Timer T1 und T2 zählen die 2 Bytes breite Vorgabe bei jedem Takt um 1 herunter. Der Timer T2 wurde durch das Betriebssystem zur Echtzeit-Uhr TI$ programmiert, während der leistungsfähigere T1 dem Benutzer voll zur Verfügung steht. Diesem sind 4 Registeradressen zugeordnet: Zwei Vorspeicher (latches) und 2 Zählregister. Das Laden der zwei Zählregister aus dem Vorspeicher geschieht erst automatisch beim Laden des niederwertigen Zählerbytes (LSB).

Alle Funktionen werden durch die Bitpositionen 7 und 6 im ACR festgelegt, hierbei ist die Besonderheit, daß der erzeugte Impuls auf PB7 (ORB-Bit 7) als Rechteckimpuls herausgeführt werden kann.

ACR7	ACR6	Bedeutung
0	0	Einzelimpuls – keine Ausgabe auf PB7
0	1	Dauerimpulse – keine Ausgabe auf PB7
1	0	Einzelimpuls – Ausgabe auf PB7
1	1	Dauerimpulse – Ausgabe auf PB7

Genau diese letzte Kombination machen wir uns zunutze. Leider ist das Signal PB7 (6522 – Pin 17) beim PET nicht nach außen geführt, was wir durch Anlöten und Herausführen eines Drahtes nachholen.

Der VIA 6522 arbeitet mit einer Taktfrequenz von 1 MHz. Die Breite des Impulses berechnet sich nach folgender Formel, wobei LSB das niederwertige und MSB das höherwertige zu ladende Byte sind:

$T = (MSB * 256 + LSB + 2) * 2$ [Mikrosekunden]

Der Faktor * 2 entsteht dadurch, daß bei Ablauf der Zeitdauer der TTL-Pegel an PB7 automatisch invertiert wird. Hieraus folgt für die Frequenz:

$F = 1/T$ [MHz]

Nach Auflösung der obigen Gleichung ergeben sich für LSB und MSB folgende BASIC-Statements:

MSB = INT ((1000000 / F –4) / 512)
LSB = INT (((1000000 / F –4) – 512 * MSB) /2)

Folgendes kleines BASIC-Programm erlaubt die Programmierung von beliebigen Rechteckimpulsen zwischen 7,66 Hz und 250 kHz:

10 POKE 59467,196: REM Timer 1 programmieren; 20 INPUT „Frequenz eingeben (Hz)"; F; 30 MSB = INT ((1000000 / F –4) / 512); 40 LSB = INT (((1000000 / F –4) – 512 * MSB) /2); 50 IF MSB > 255 OR LSB > 255 THEN PRINT „Nicht darstellbar!": GOTO 20; 60 POKE 59461, MSB: REM Timer laden mit MSB; 70 POKE 59460, LSB: REM Timer laden mit LSB; 80 GOTO 20.

Die programmierte Frequenz bleibt so lange eingestellt (free running mode), bis eine neue Programmierung erfolgt. Vor Kassettenoperationen ist das ACR-Register mit POKE 59467,0 wieder zu löschen. Klaus Musseleck

Repeat-Funktion für die CBM-Tastatur

Das im *Bild* disassembliert aufgelistete Programm findet im Kassettenbuffer-2-Speicherbereich der CBM-Computerserie Platz und ermöglicht eine Repeat-Funktion aller Tasten (außer Shift natürlich). Die Repeat-Funktion läßt sich mit SYS 826 ein- und ausschalten. Zu beachten ist allerdings, daß die Repeat-Funktion auch dann abgeschaltet wird, wenn der Interrupt-Vektor anderweitig verändert wird, z. B. beim Laden von der Kassette.

Ist das Programm aktiv, so wird jede Taste, die länger als eine Sekunde gedrückt ist, mit einer Geschwindigkeit von zehn Zeichen pro Sekunde wiederholt (diese Zeitwerte lassen sich in den Speicherzellen hex 0354 und 0362 verändern).

Dr. Harald Lindner

```
033A 78        SEI              ;Vertauschung der Interrupt-
033B A5 90     LDA 90           ;vektoren E62E und 0349
033D 49 67     EOR #67
033F 85 90     STA 90
0341 A5 91     LDA 91
0343 49 E5     EOR #E5
0345 85 91     STA 91
0347 58        CLI
0348 60        RTS
0349 A6 0F     LDX 0F           ;Alte Taste (selbst gemerkt)
034B A5 97     LDA 97           ;Alte Taste (CBM)
034D 85 0F     STA 0F           ;selber merken
034F C9 FF     CMP #FF          ;Keine Taste gedrueckt?
0351 F0 12     BEQ 0365         ;Dann --->
0353 A9 3C     LDA #3C          ;60 Jiffies = 1 Sekunde
0355 E4 97     CPX 97           ;Ist noch dieselbe Taste gedrueckt?
0357 D0 0A     BNE 0363         ;Nein: Dann Zaehler 1 Sekunde setzen
0359 C6 10     DEC 10           ;Zaehler dekrementieren
035B D0 08     BNE 0365         ;Noch nicht 0: Dann --->
035D A9 FF     LDA #FF          ;Mache CBM glauben, dass vorher
035F 85 97     STA 97           ;keine Taste gedrueckt war
0361 A9 06     LDA #06          ;Zaehler 1/10 Sekunde setzen
0363 85 10     STA 10
0365 4C 2E E6  JMP E62E         ;Normalen Interrupt fortsetzen
```